Constrained Delaunay Tetrahedralizations, Bistellar Flips, and Provably Good Boundary Recovery

Jonathan Richard Shewchuk
University of California at Berkeley
Berkeley, California
jrs@cs.berkeley.edu

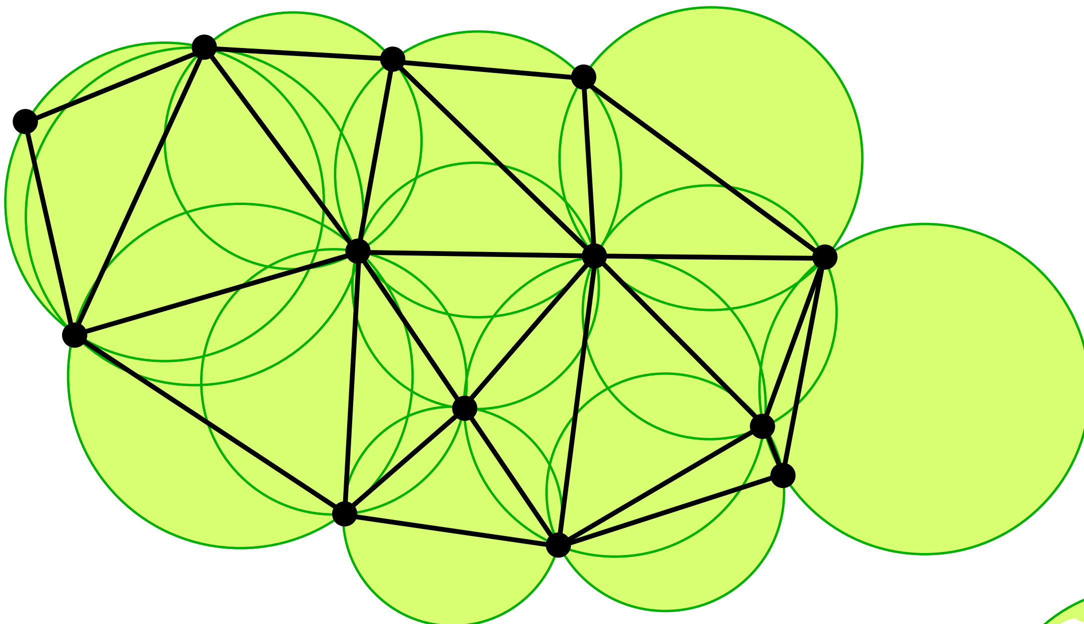
Delaunay Triangulations: Just Dandy

Every set of vertices, in any dimension, has one.

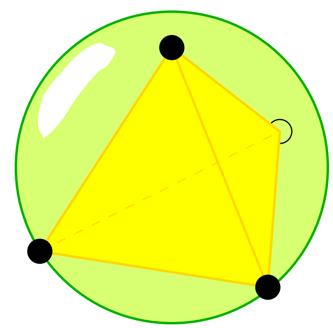
- ‘‘Nicely’’ shaped triangles, tetrahedra, etc.
- The ‘‘optimal’’ triangulation by several criteria.
- Great for interpolation: for a bounded−curvature function f with piecewise linear interpolant g, Delaunay minimizes the worst−case bound on the interpolation error $\| f - g \|_\infty$.
- About one in twelve papers in computational geometry concern Delaunay triangulations.

Delaunay Triangulations

- The circumscribing circle of every Delaunay triangle is empty.

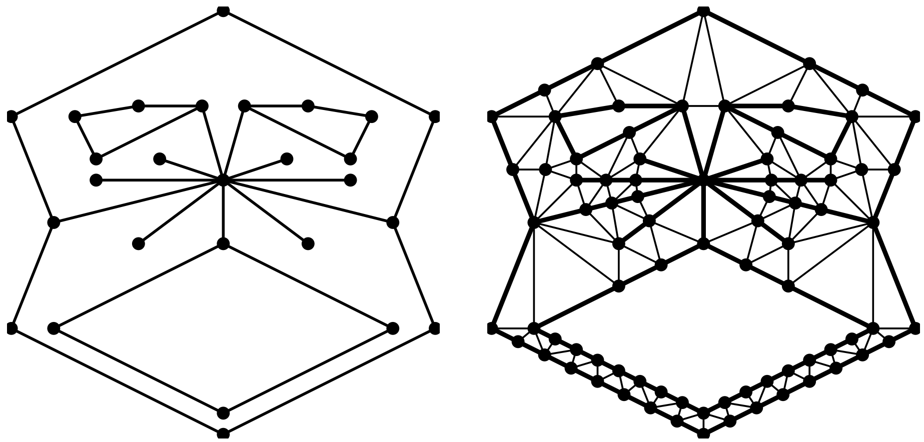


- The Delaunay triangulation generalizes to higher dimensions.

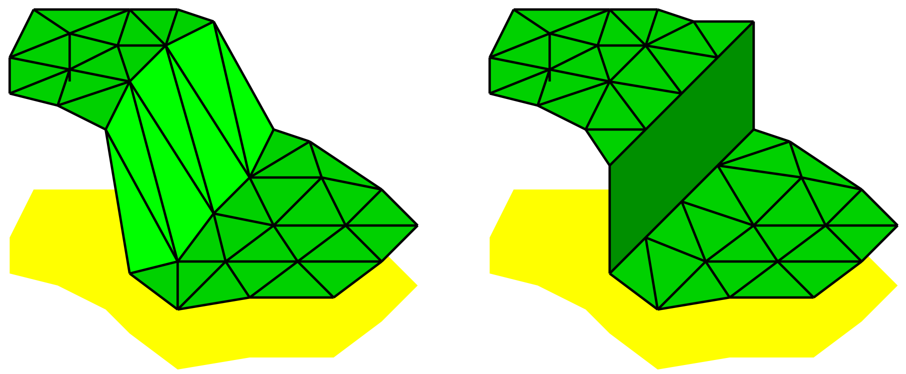


Delaunay triangulations are great, but sometimes you need to make sure edges or facets appear.

- Nonconvex shapes; internal boundaries

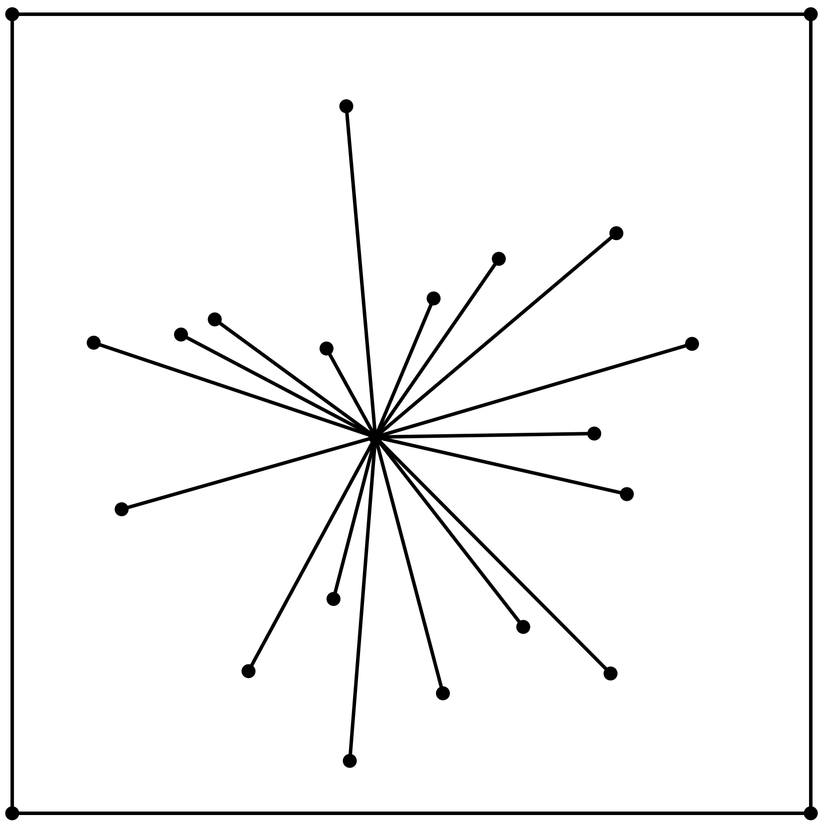


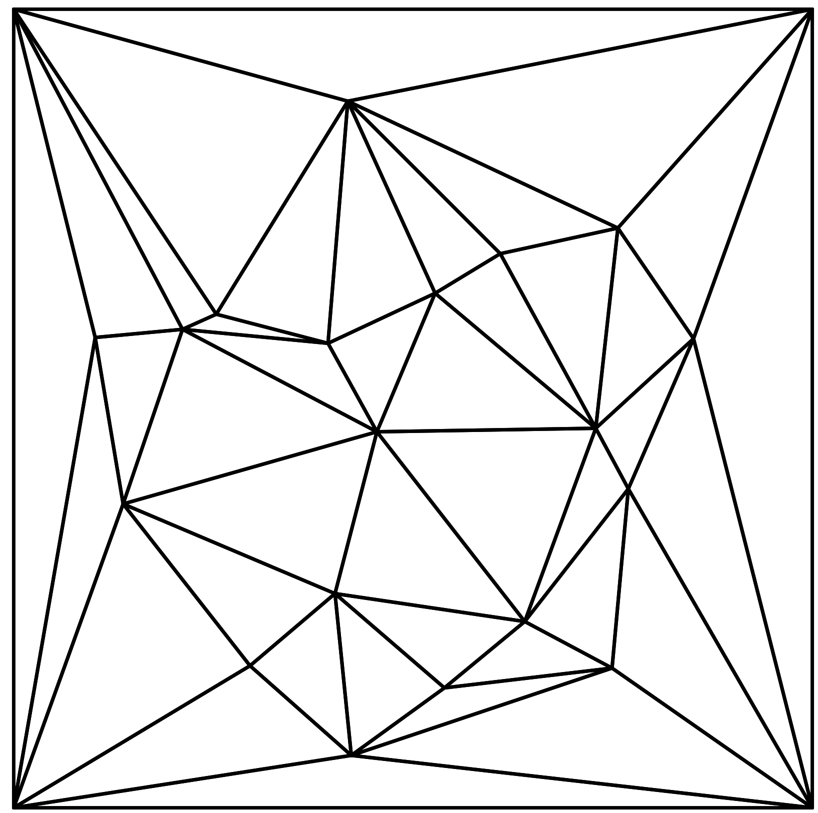
- Discontinuities in interpolated functions



Typical input:

Planar Straight−Line Graph (PSLG).

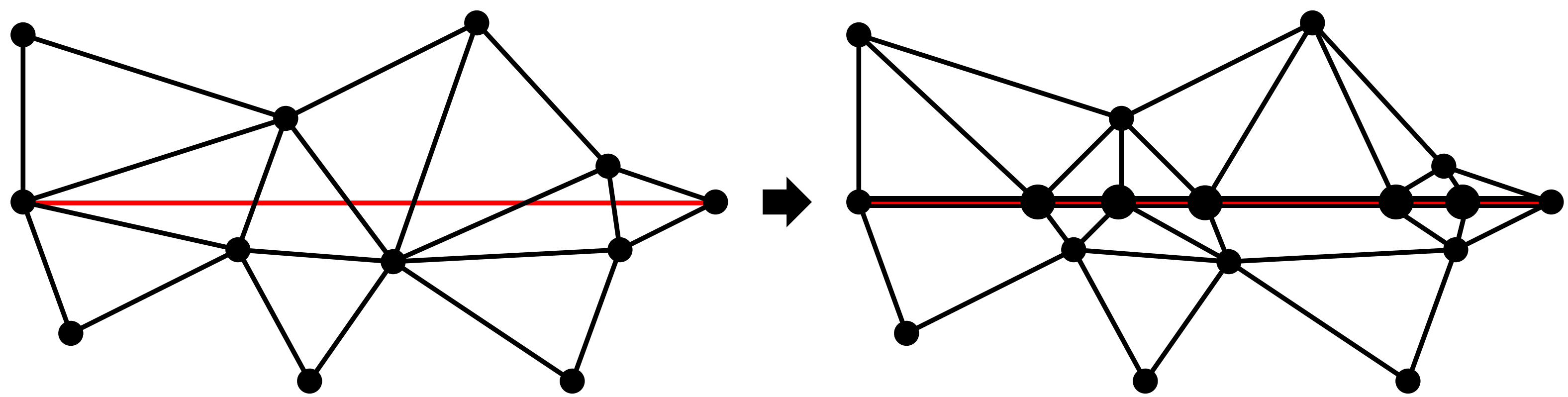




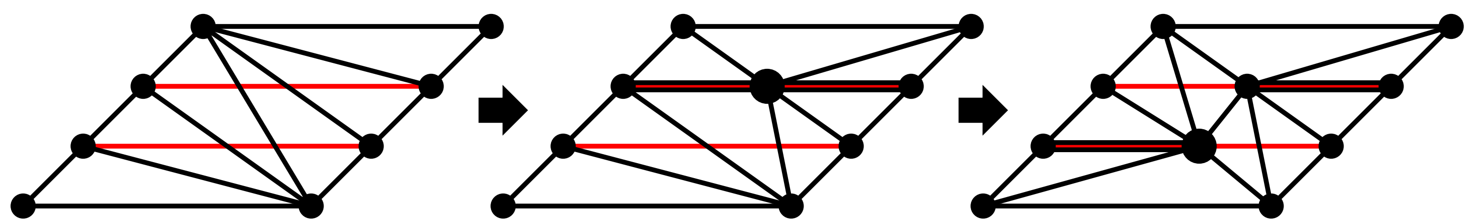
Delaunay triangulation of vertices might not conform to constraining segments.

Conforming Delaunay Triangulations

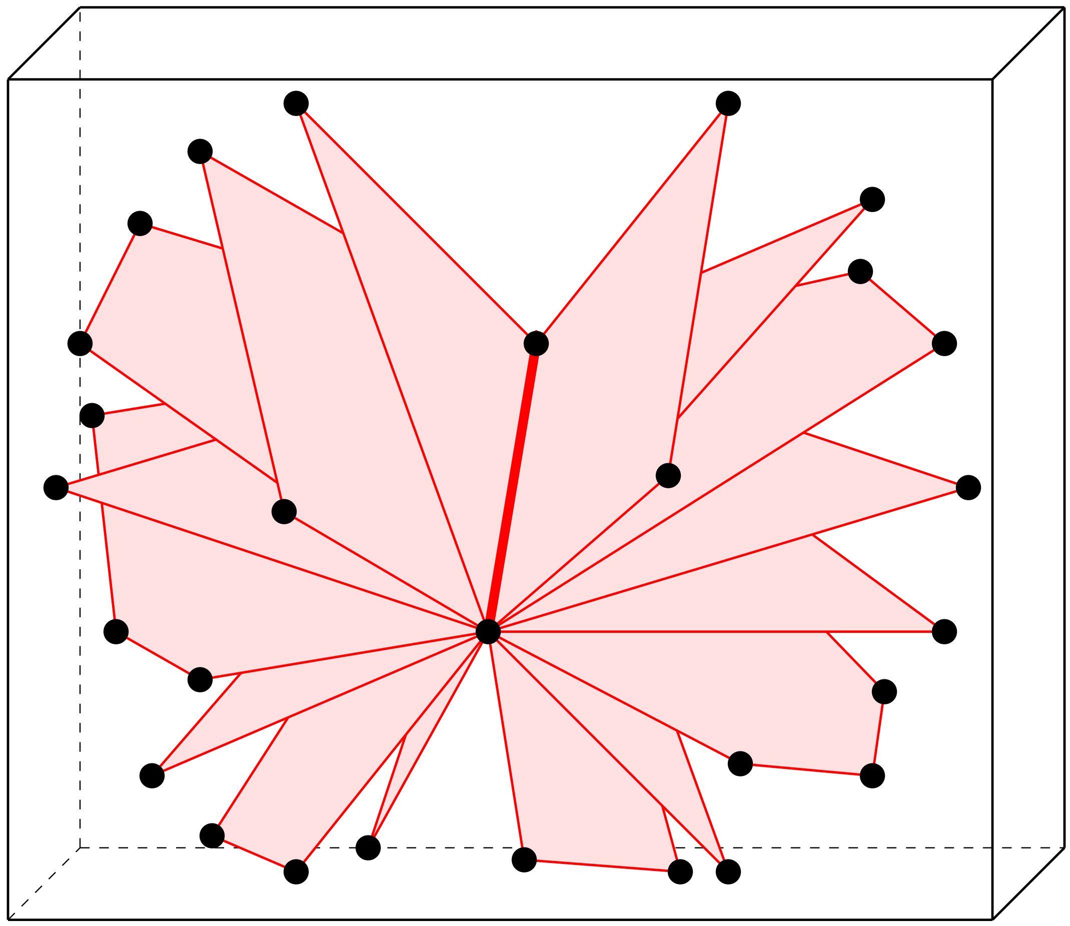
- Missing segments can be recovered by adding vertices...



- ...but they can also be knocked out!



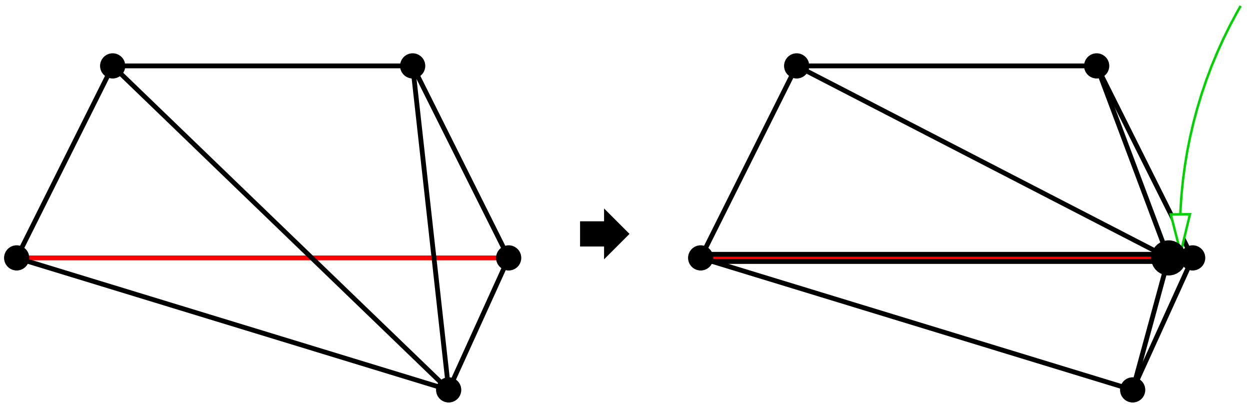
A Hard Example for Tetrahedral Meshing



It is difficult to mesh the interior of this box with Delaunay tetrahedra. A new vertex inserted in one facet tends to knock out triangular faces in adjacent facets.

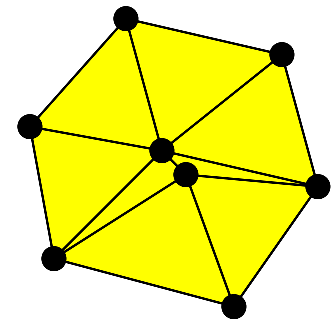
Conforming Delaunay Triangulations: Algorithms

- 2D: Edelsbrunner & Tan [1993].
  - May add up to a cubic number of new vertices.
  - Triangulation edges may be arbitrarily small.

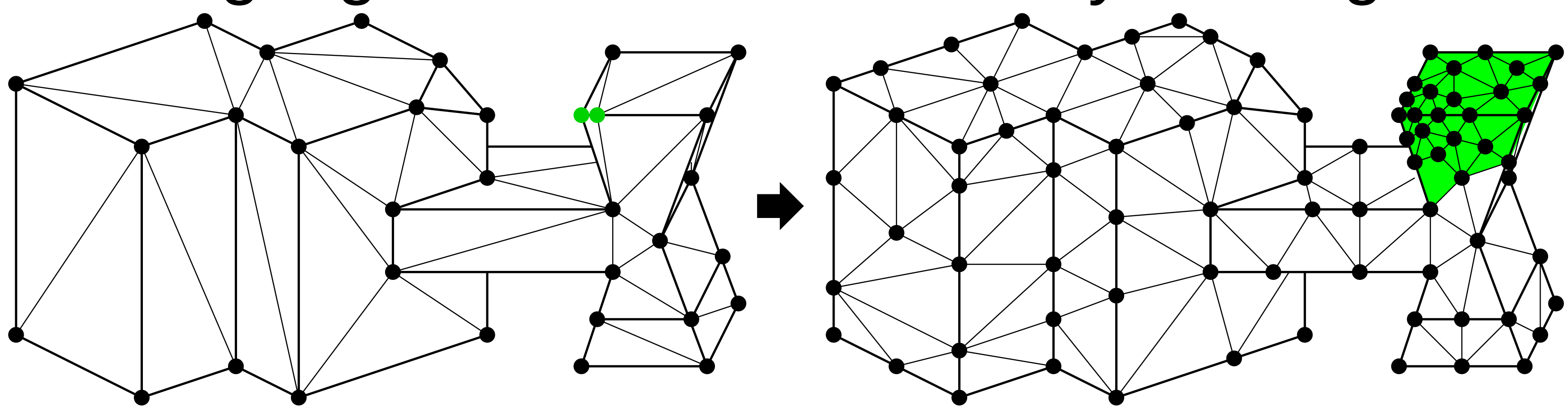


- 3D: Murphy, Mount, & Gable [2000]; Cohen−Steiner, Colin de Verdière, & Yvinec [2002].
  - No polynomial bound on # of new vertices.
  - Triangulation edges may be arbitrarily small.

Edge Lengths are More Important than a Polynomial Bound

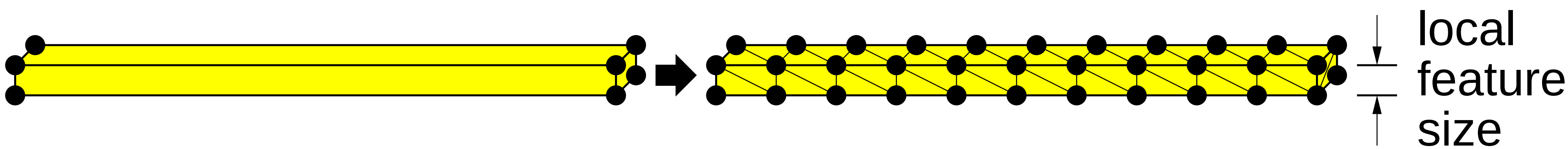


- Disparate edge lengths usually cause difficulties with interpolation & conditioning.
- Meshing algorithms can fix them by refining...



...possibly at great cost!

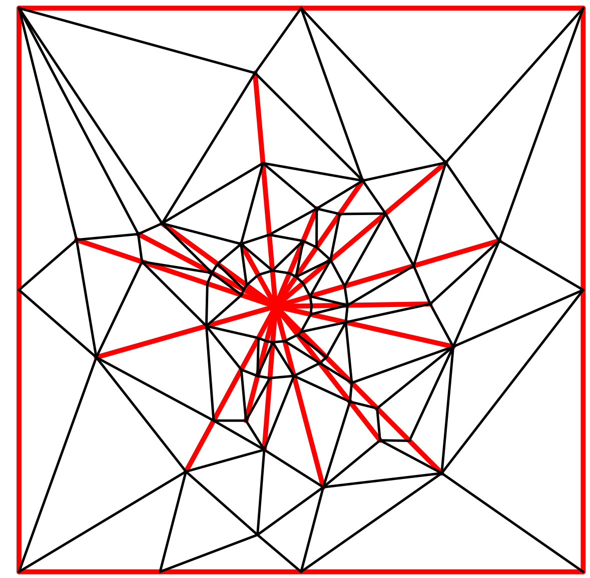
- Conversely, a polynomial bound isn’t always possible for good−quality tetrahedra.



Three Alternatives for Enforcing Constraints

Conforming Delaunay triangulations

- Tetrahedra, triangles, & edges are all Delaunay.

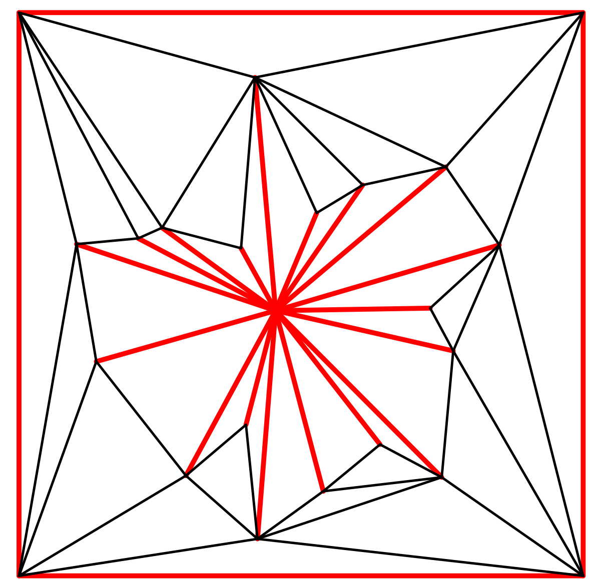


“Almost Delaunay” triangulations

- Delaunay property compromised to recover boundary facets.
- What most 3D Delaunay meshing algorithms do.

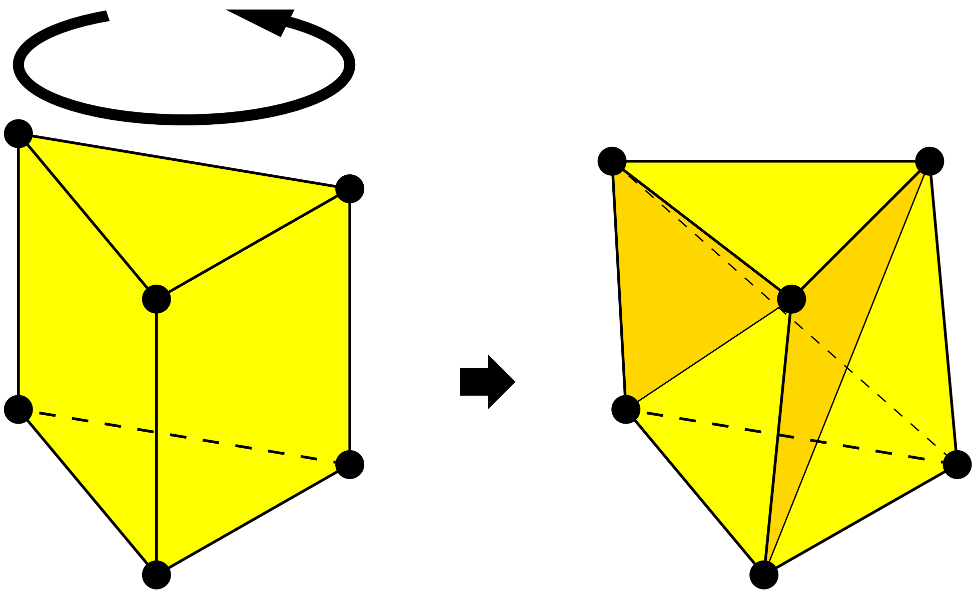
Constrained Delaunay triangulations (CDTs)

- Tetrahedra, triangles, & edges are constrained Delaunay or are domain boundaries.



Delaunay triangulations exist in all dimensions.

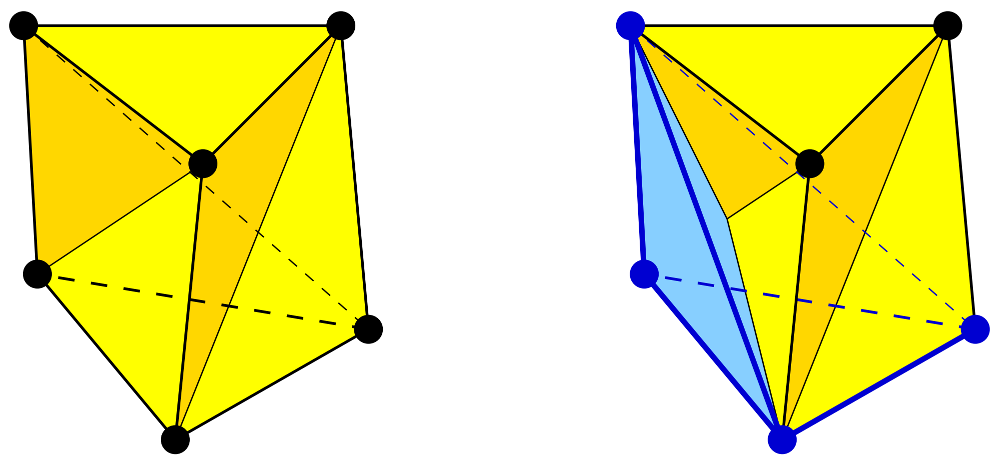
Why haven't CDTs been generalized beyond E^2?



Schönhardt's polyhedron

One reason: not every polyhedron can be tetrahedralized without extra vertices.

Not every polyhedron can be tetrahedralized without extra vertices.



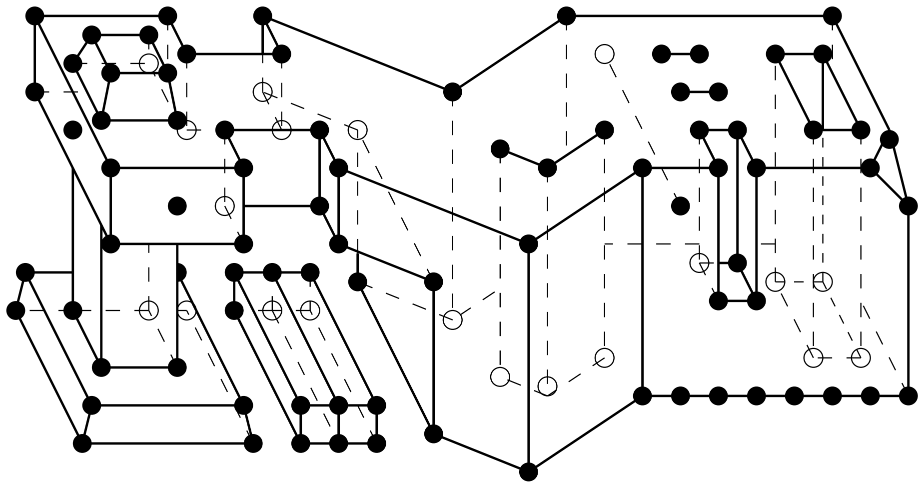
Any four vertices of Schönhardt’s polyhedron yield a tetrahedron that sticks out a bit.

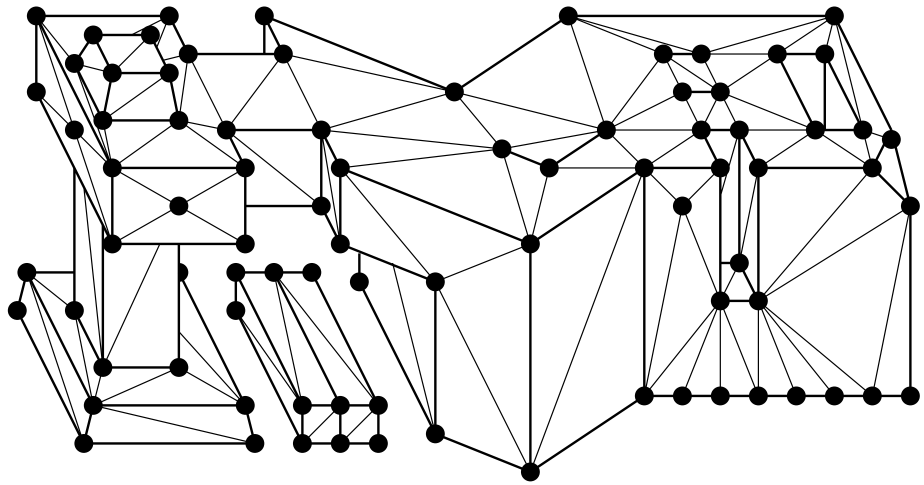
In this talk, I generalize CDTs to E^3 anyway.

Why Choose Constrained Delaunay?

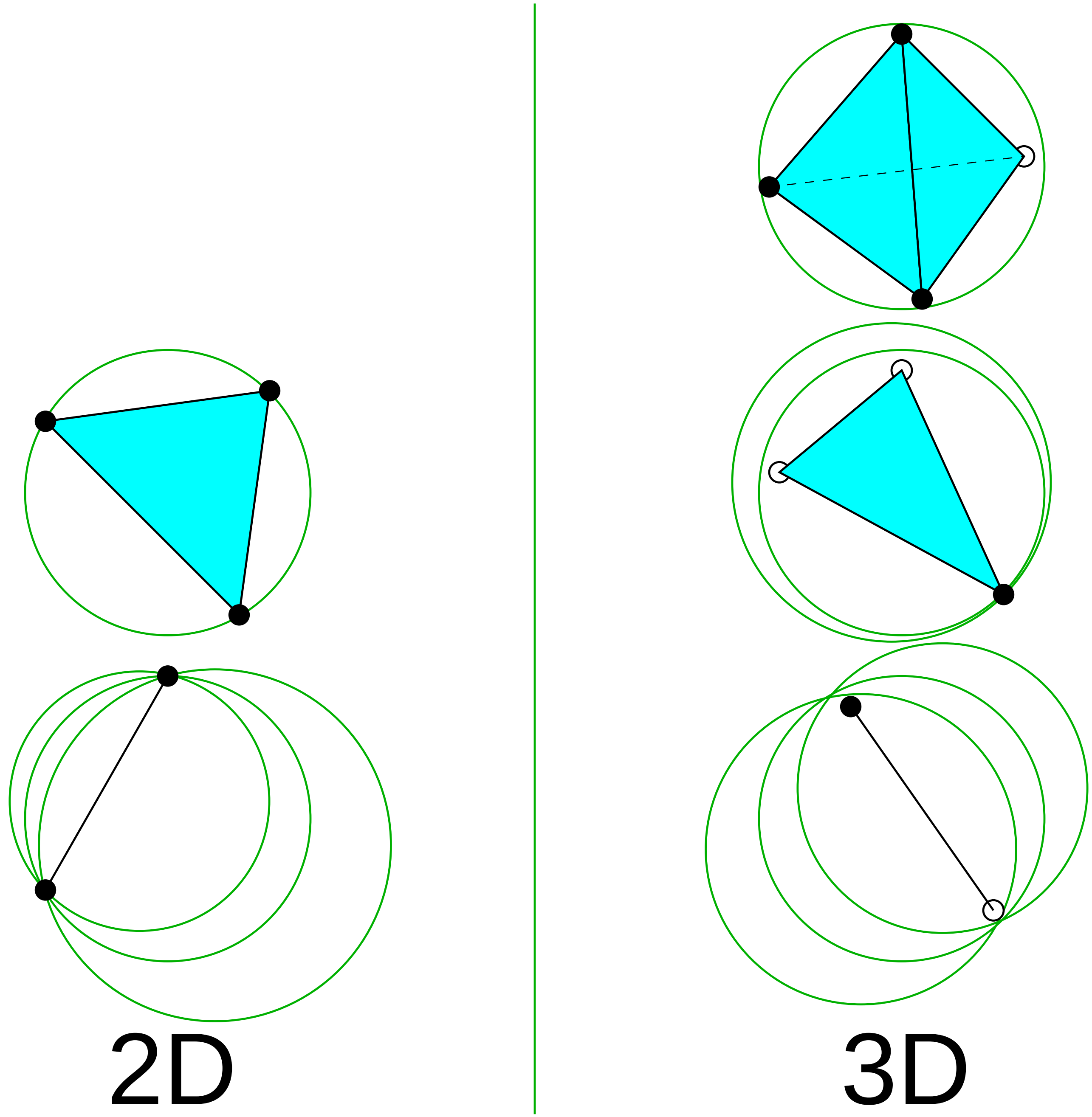
- Requires fewer new vertices than the other options.
- Optimal for minimizing the worst−case bound on the interpolation error $\| f - g \|_\infty$.
- Works in concert with Delaunay refinement algorithms to offer provably good mesh generation.
- Offers guaranteed lower bounds on edge lengths: provably good boundary recovery.

I. Constrained Delaunay Tetrahedralizations

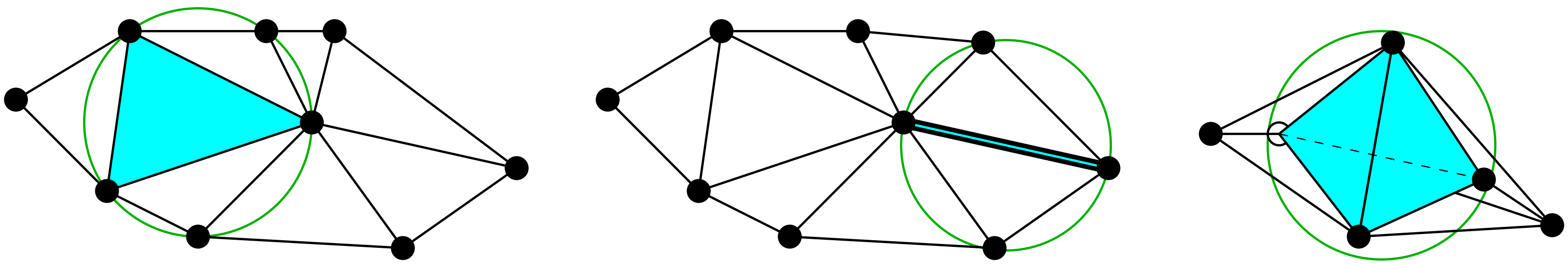




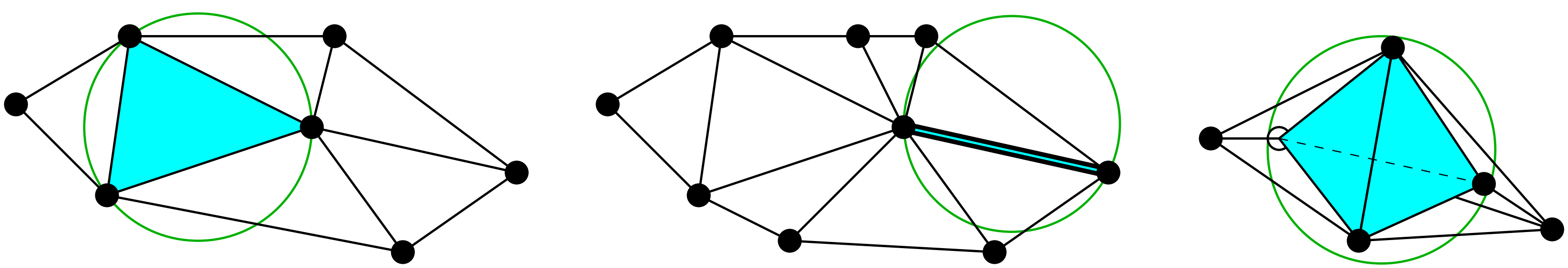
Circumsphere = Circumscribing Sphere



Simplex is *Delaunay* if no vertices inside circumsphere.

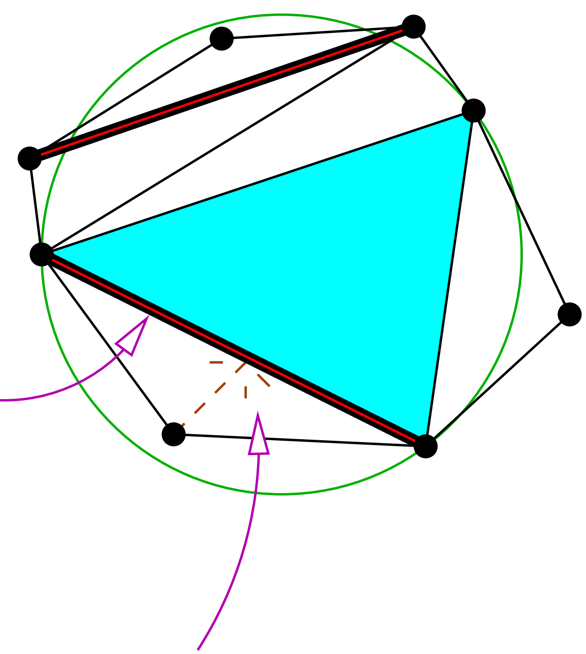


Simplex is *strongly Delaunay* if no vertices inside **or on** circumsphere, except simplex vertices.



In 2D, a triangle is *constrained Delaunay* if

- interior of triangle doesn't intersect any input segment, and
- no vertex inside triangle's circumcircle is visible from interior of triangle.

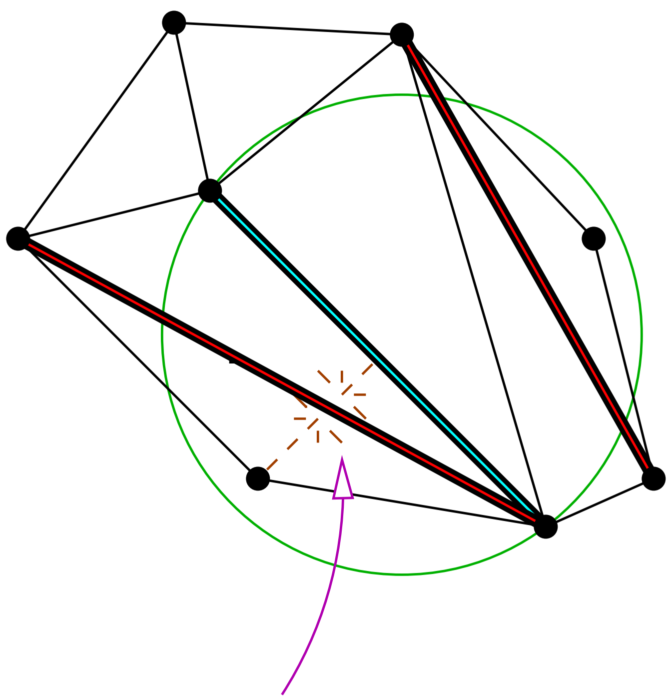


Input segments are red.

Segment occludes visibility between vertex and triangle.

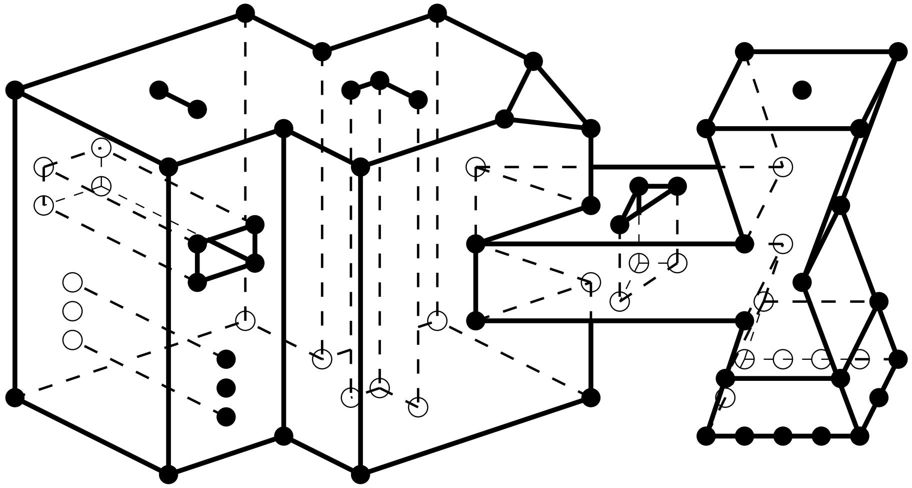
Edge is *constrained Delaunay* if

- Edge doesn’t cross any input segment, and
- Some circumcircle encloses no vertex visible from interior of edge.



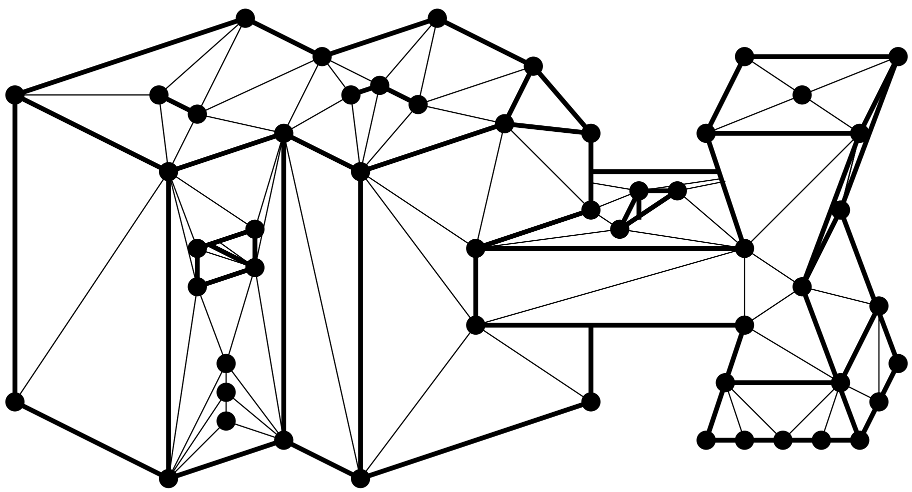
Segment occludes visibility between vertex and edge.

Input: A Piecewise Linear Complex



PLC

Set of vertices, *segments,* and *facets.*

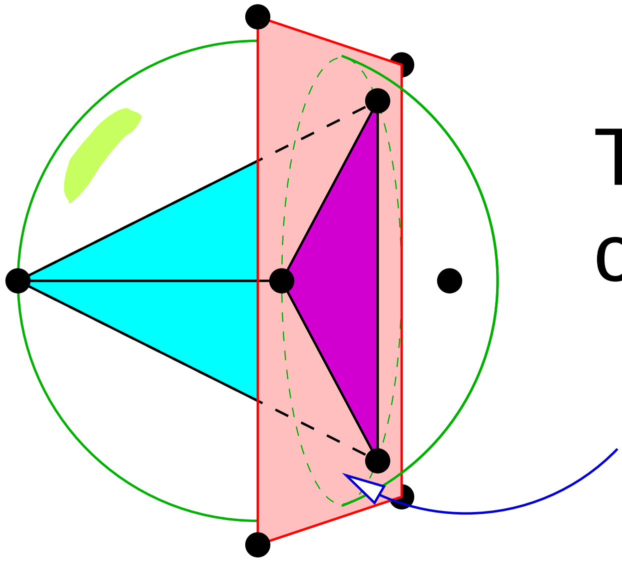


CDT

Each facet appears as a union of triangular faces.

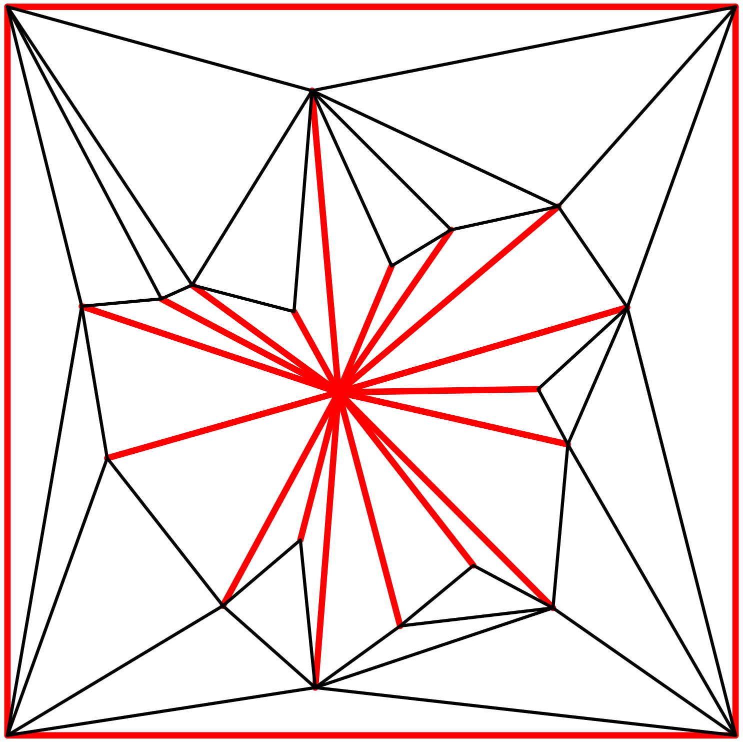
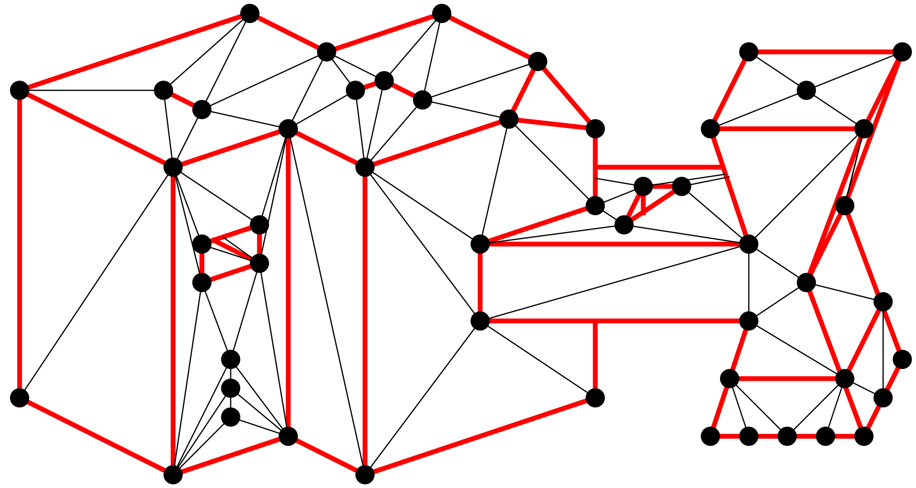
What is “constrained Delaunay” in 3D?

Same as 2D, but only constraining facets of the PLC occlude visibility.



The blue tetrahedron is constrained Delaunay.

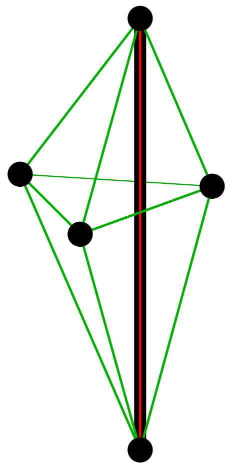
Constraining facet hides vertex

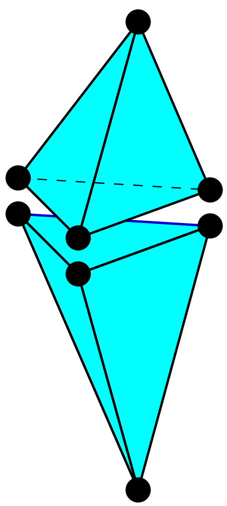
The constrained Delaunay triangulation is the union of all constrained Delaunay simplices, *if* the union is a triangulation. In 3D, a PLC might have no CDT.

What is “constrained Delaunay” in 3D?

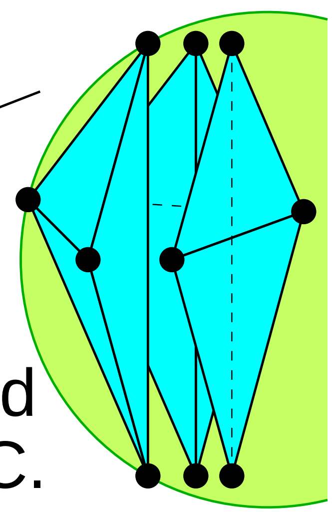
Segments do not affect visibility.



Doesn’t have a CDT.



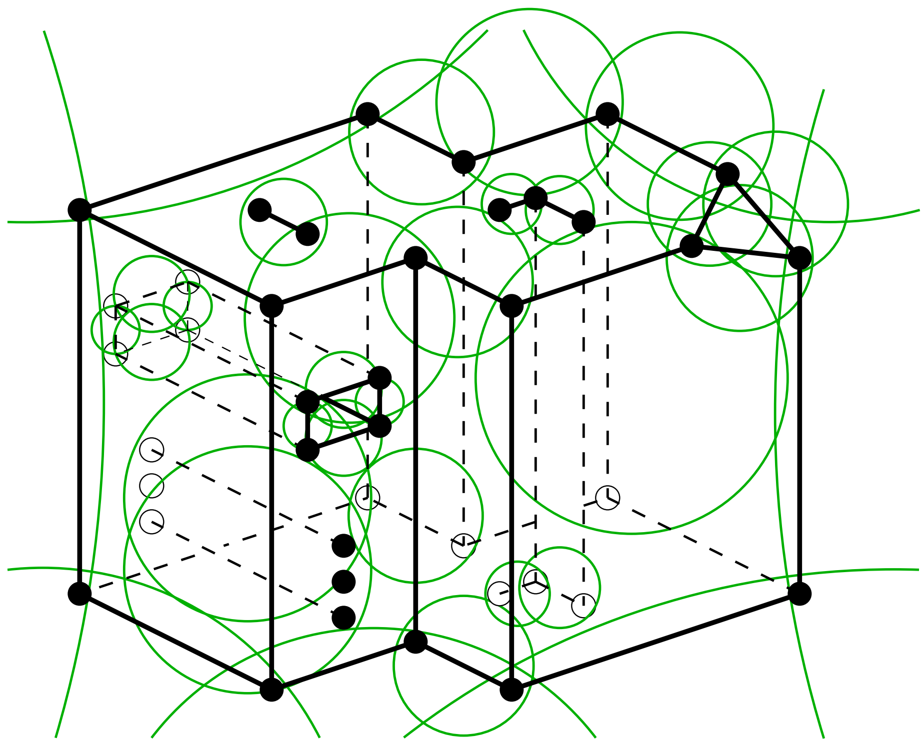
Delaunay tetrahedralization
of the vertices.



Sole constrained
tetrahedralization of the PLC.

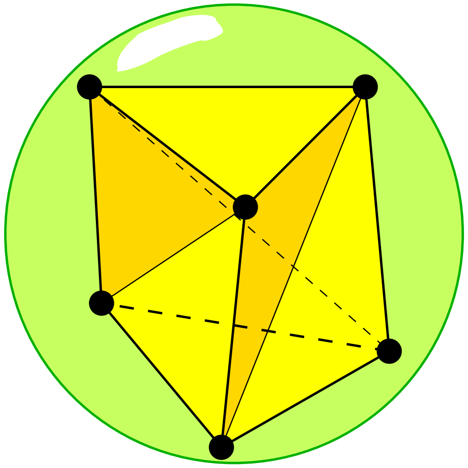
CDT Theorem (makes 3D CDTs useful)

Say that a PLC is *edge−protected* if all its segments are strongly Delaunay.

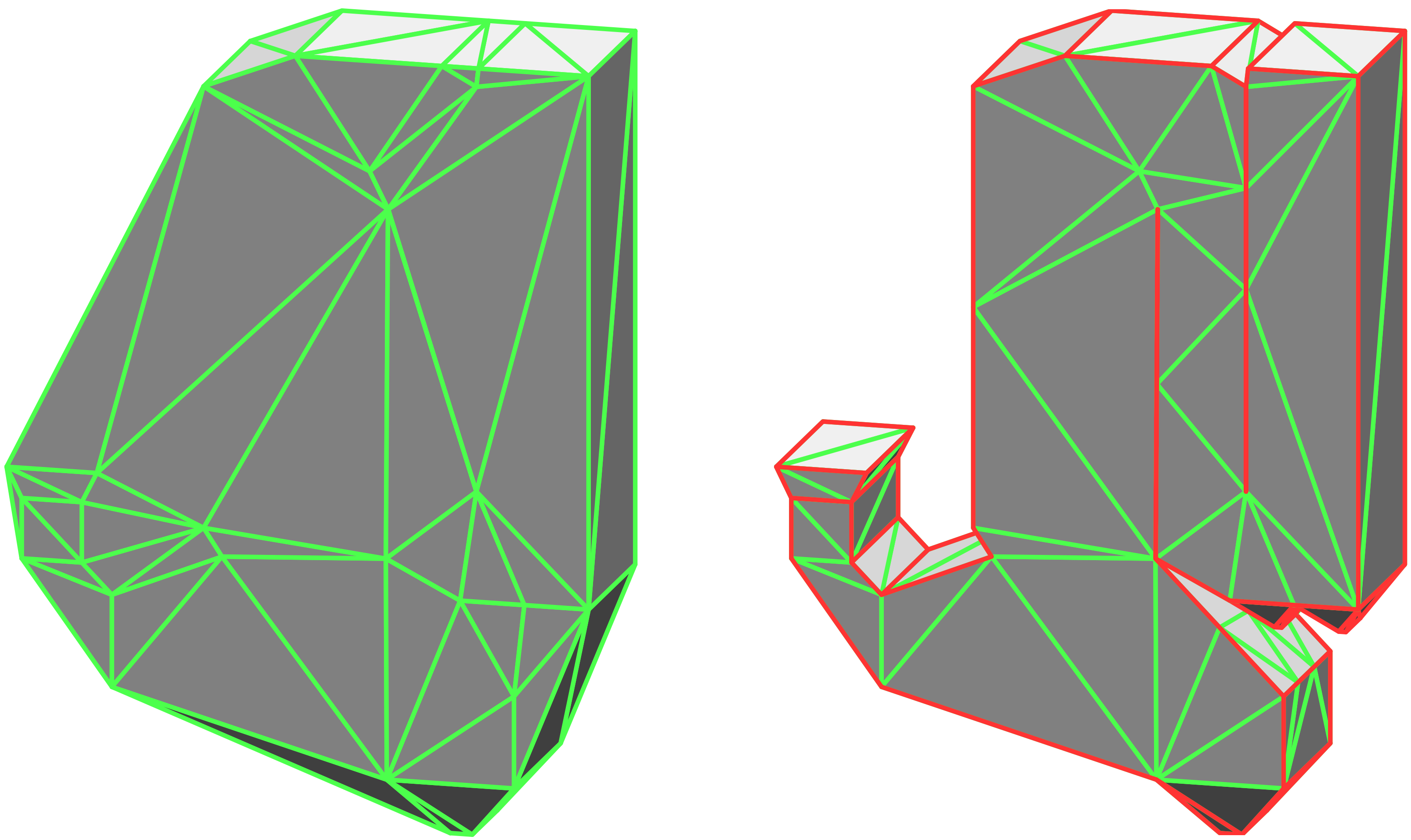


Theorem:
Every edge−protected PLC has a CDT.

Delaunay isn’t good enough.

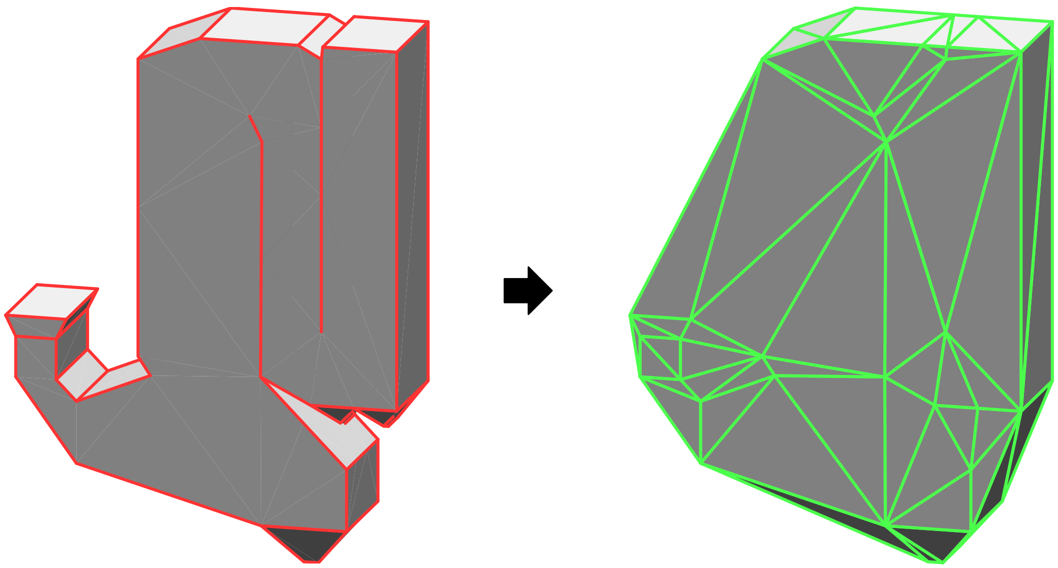


II. Provably Good Boundary Recovery



How to Tell if a PLC is Edge−Protected

- Form the Delaunay tetrahedralization of the vertices of the PLC.

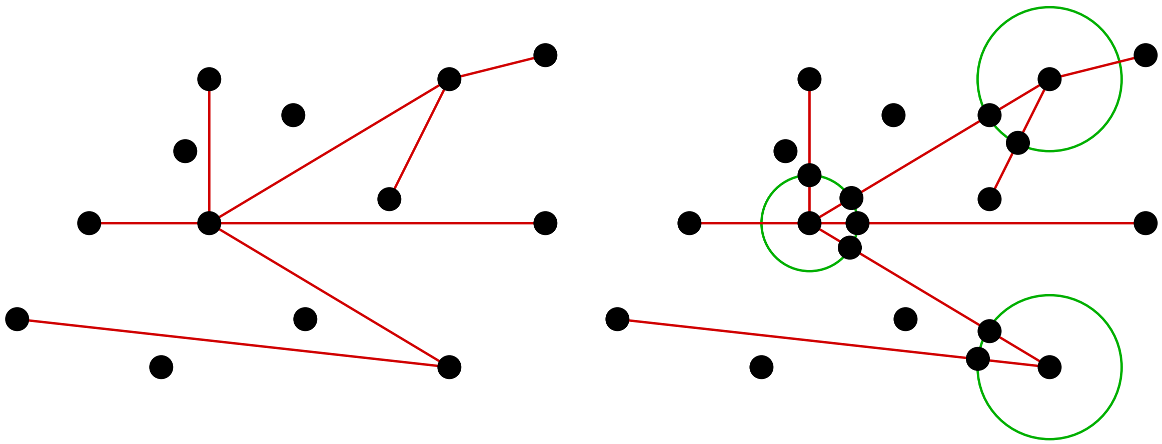


- If a segment is not in the DT, it's not strongly Delaunay.

How to Make a 3D PLC Edge−Protected

(Here demonstrated in 2D)

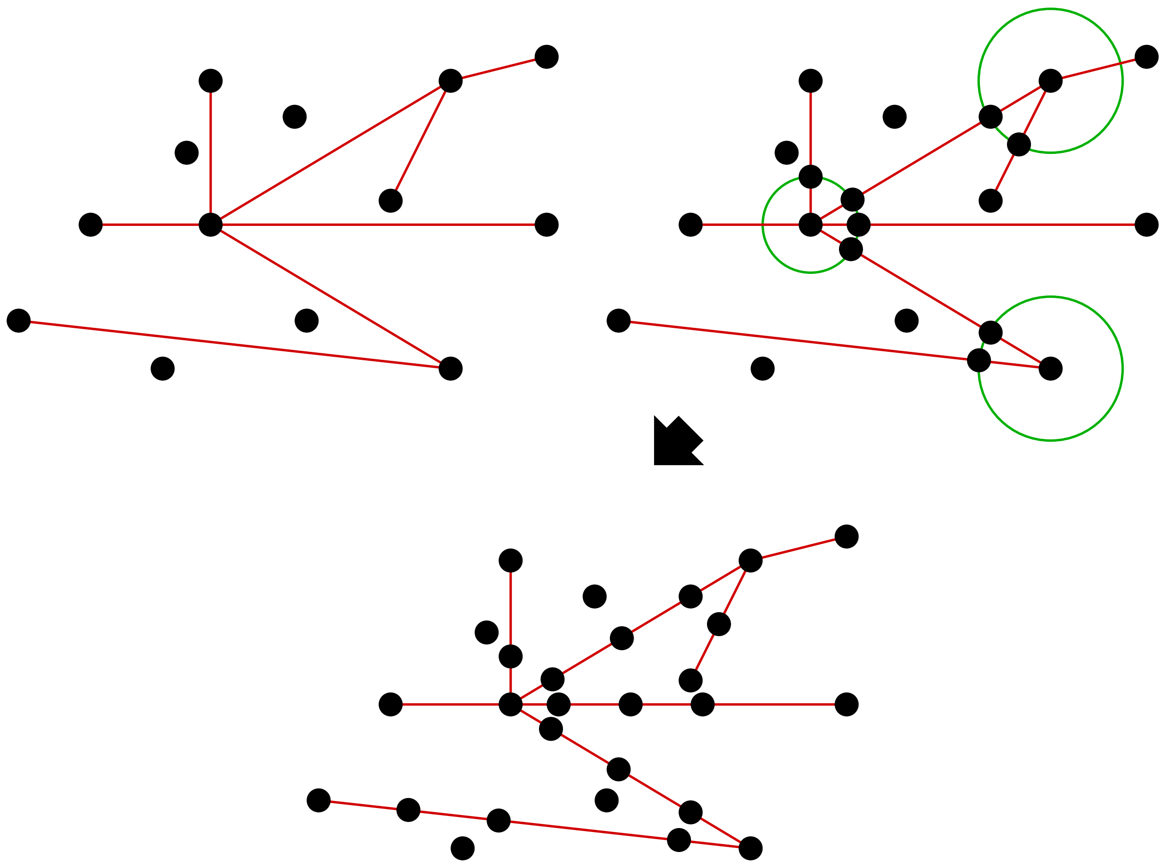
1. Subdivide segments that are incident to other segments at angles less than 90° . Use spheres of appropriate radii to cut.



How to Make a 3D PLC Edge−Protected

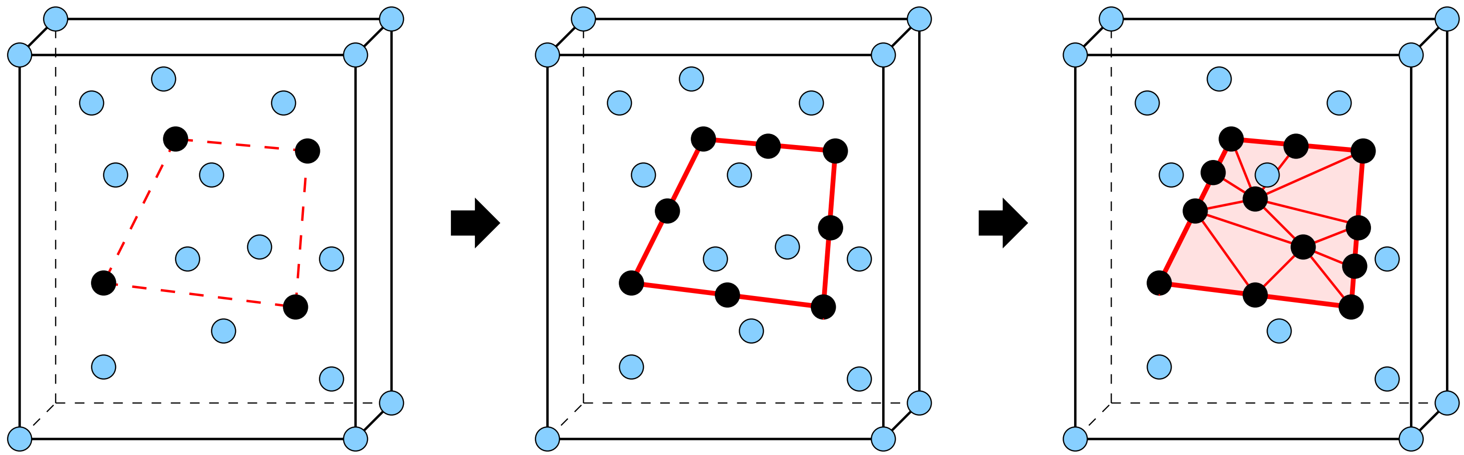
(Here demonstrated in 2D)

2. Recursively bisect any subsegment that still isn't strongly Delaunay.

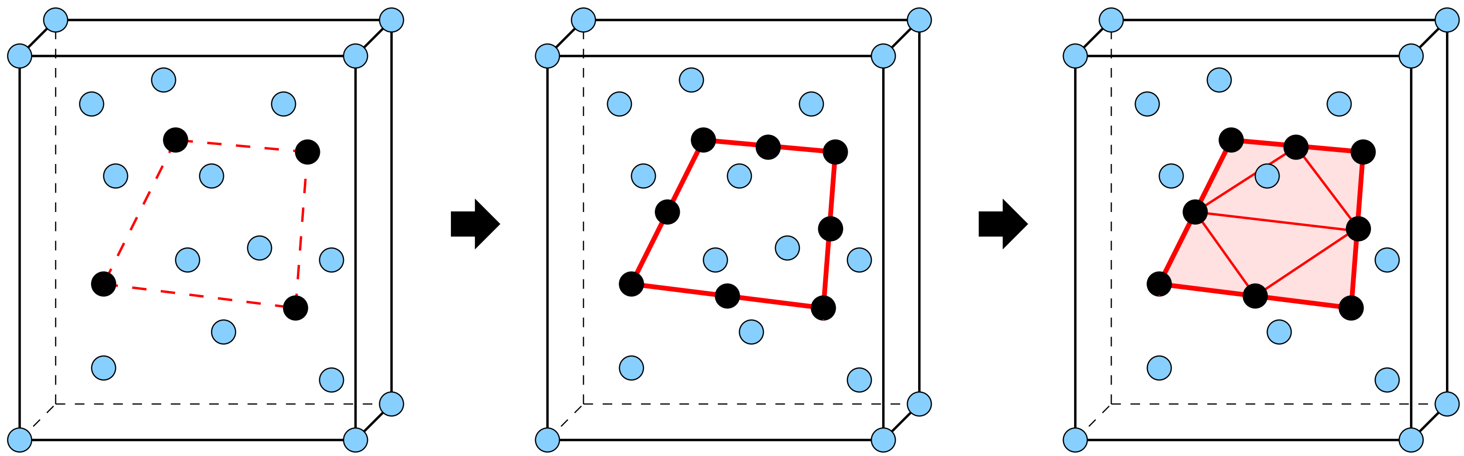


CDTs Make Shape Tetrahedralization Easier

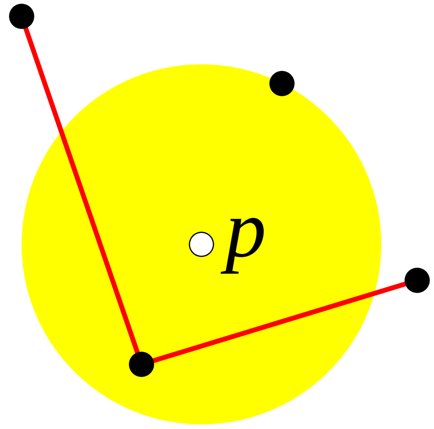
Usual Practice (Conforming Delaunay/ Almost Delaunay)



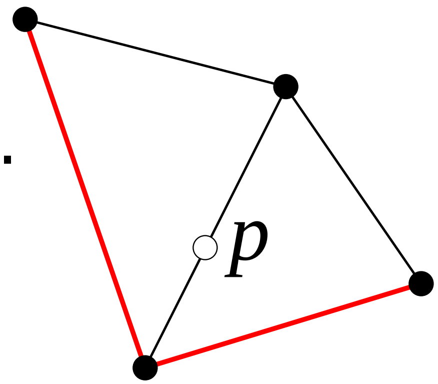
New Practice (Constrained Delaunay)



Provably Good Boundary Recovery
(Main Result)

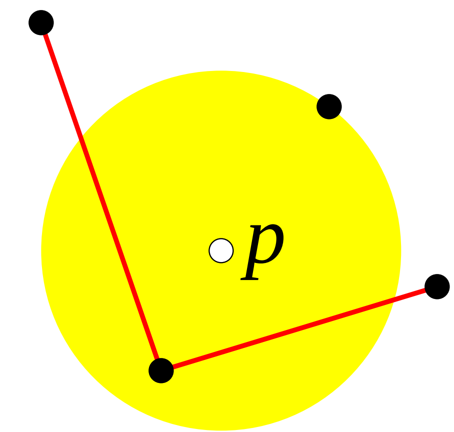


- Define *local feature size* lfs(p) at point p to be radius of smallest ball centered at p that intersects two vertices/segments that don't intersect each other.
- Any edge in final CDT has length at least lfs(p) / 4, where p is any point in the edge.



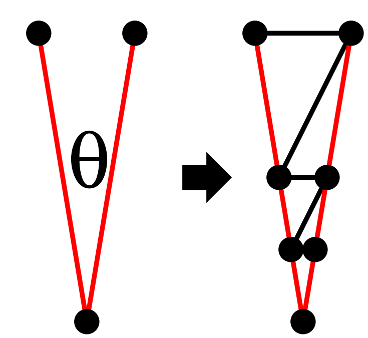
Provably Good Boundary Recovery
(Main Result)

- Define *local feature size* lfs(p) at point p to be radius of smallest ball centered at p that intersects two vertices/segments that don't intersect each other.

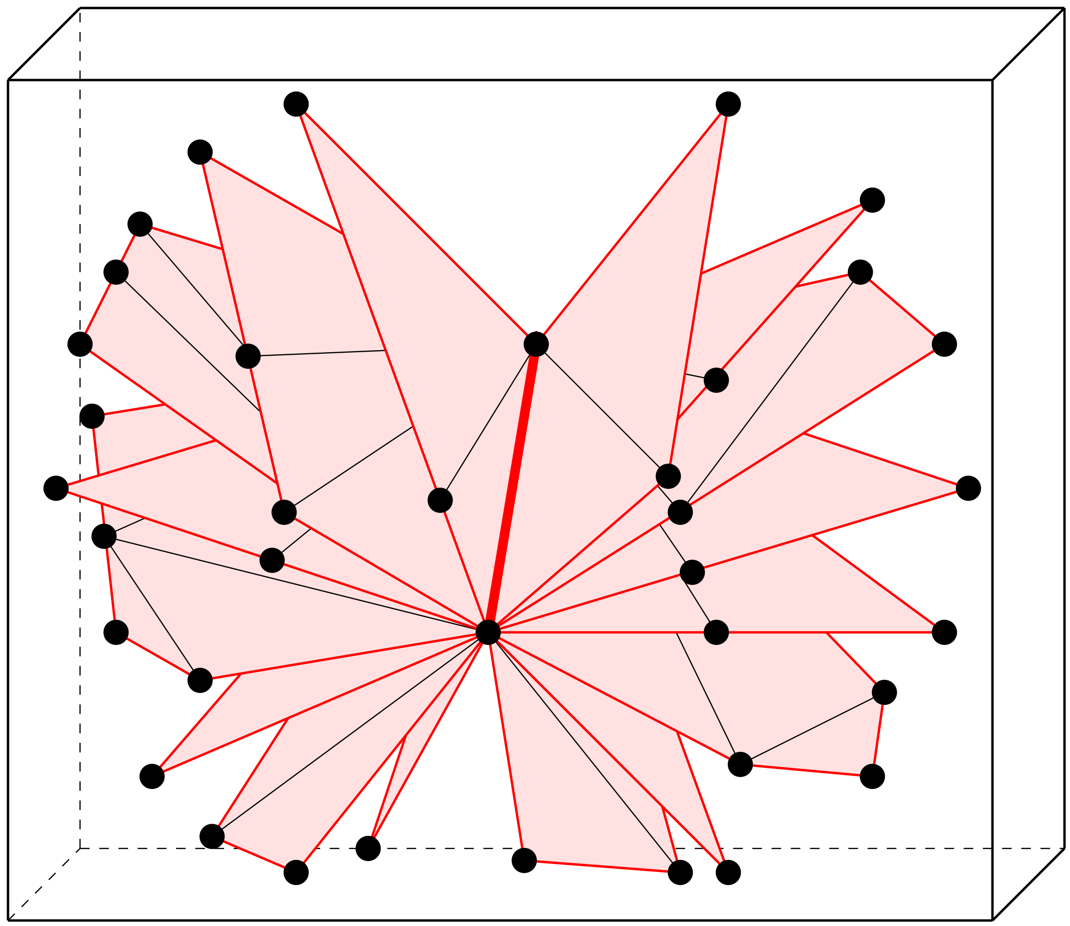


- Any edge in final CDT has length at least lfs(p) / 4, where p is any point in the edge, except...

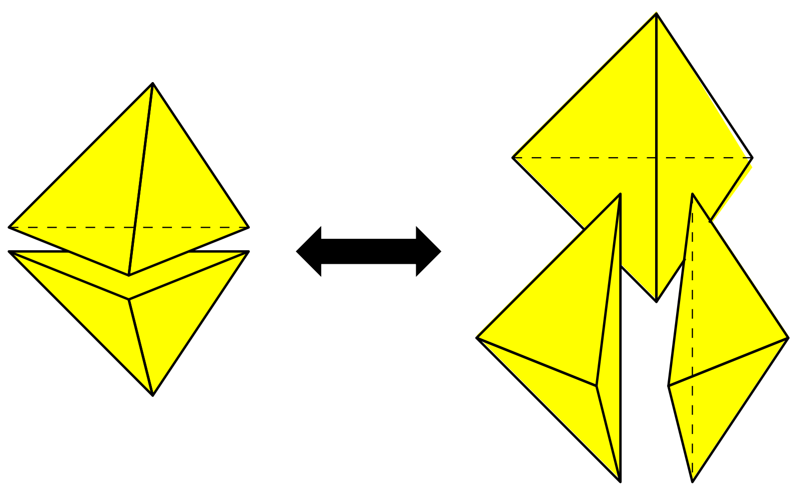
- If two intersecting input segments are separated by angle θ less than 60°, any edge with one endpoint on each segment has length at least lfs(p) sin(θ / 2) / 2.



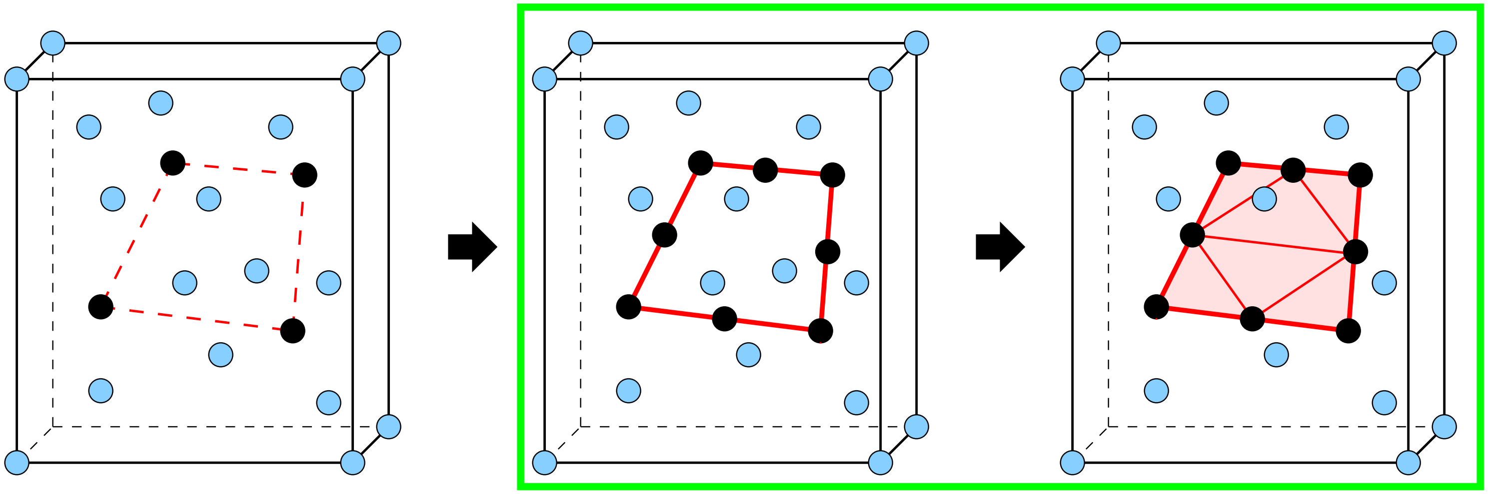
A Hard Example for Tetrahedral Meshing



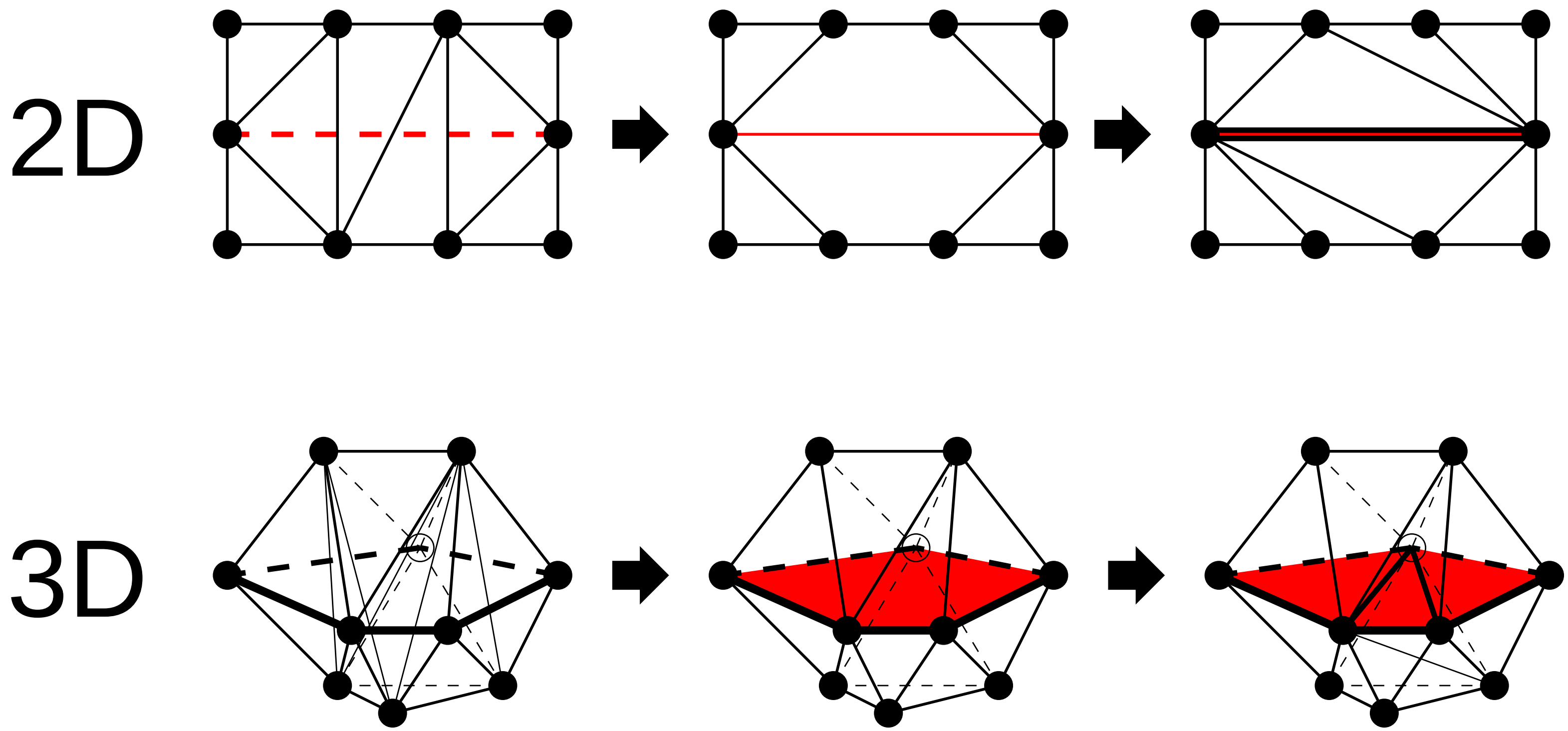
III. Bistellar Flips



How to Recover the Missing Facets?



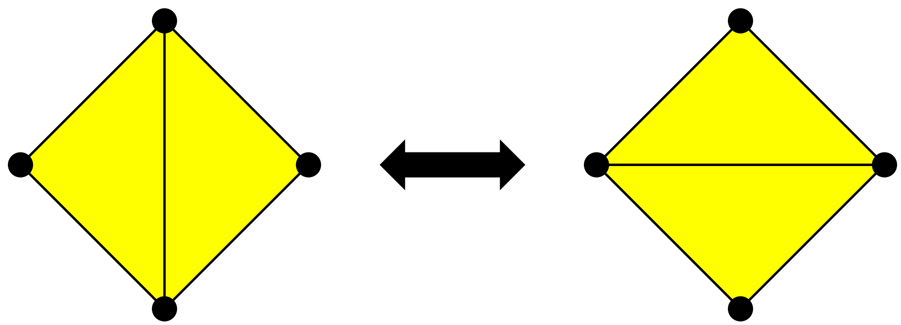
How to Recover a Facet

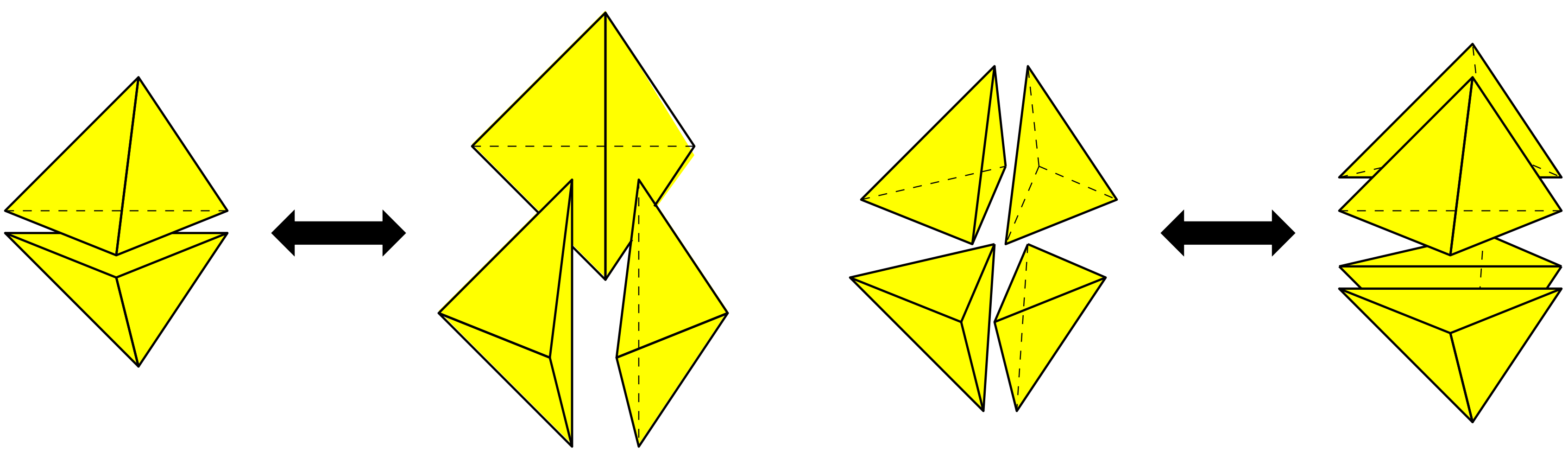


(facet must be edge−protected)

Insert the facets one by one.

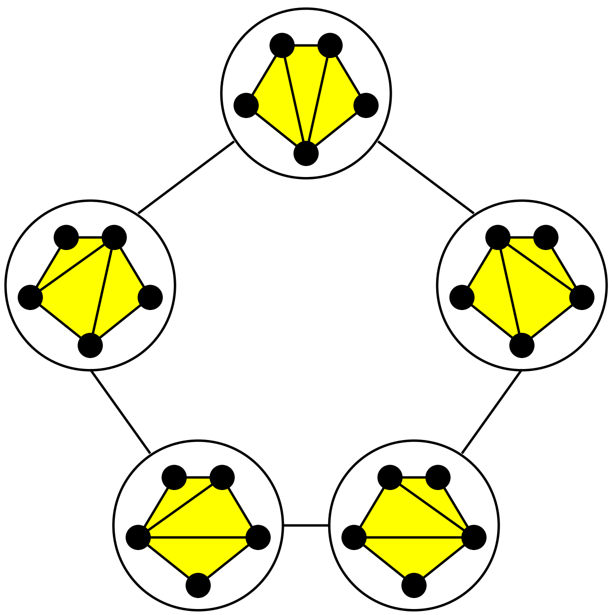
Algorithm: Use a Sequence of *Bistellar Flips*





An Aside on Flip Graphs

The flip graph of a vertex set:
a node for each triangulation; an edge for each flip.



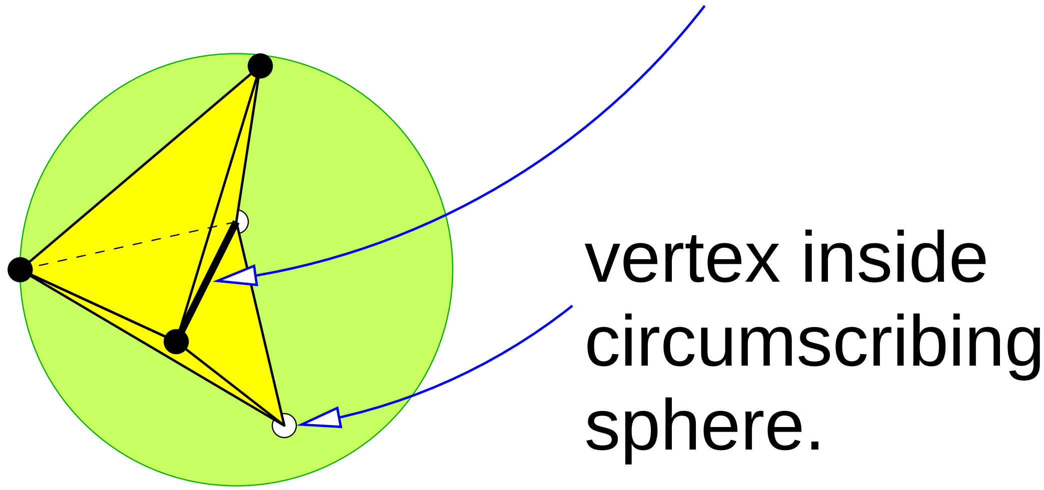
2D: Every flip graph is connected, including any subgraph induced by a PSLG.

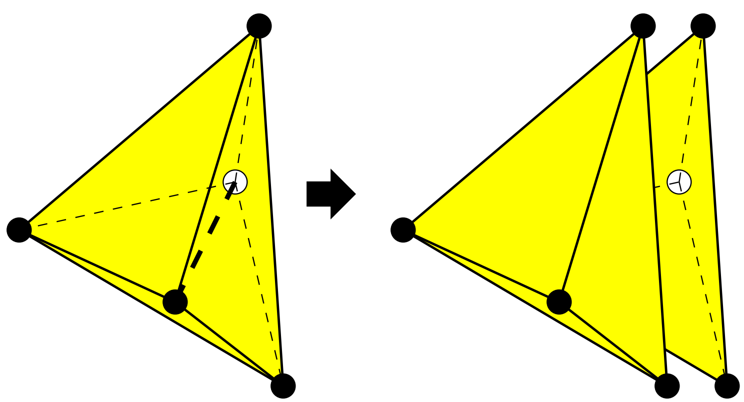
3D: Nobody knows whether or not every flip graph is connected.

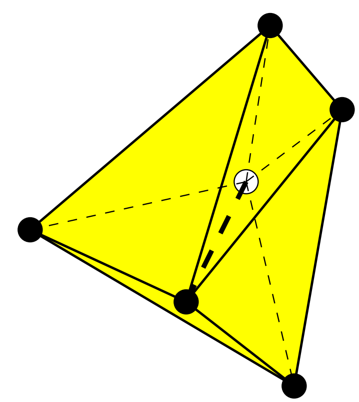
6D: Some vertex sets have flip graphs with isolated nodes [Santos 2000].

3D Delaunay Flips Can Get "Stuck"

If an edge is not Delaunay, we want to flip it away.



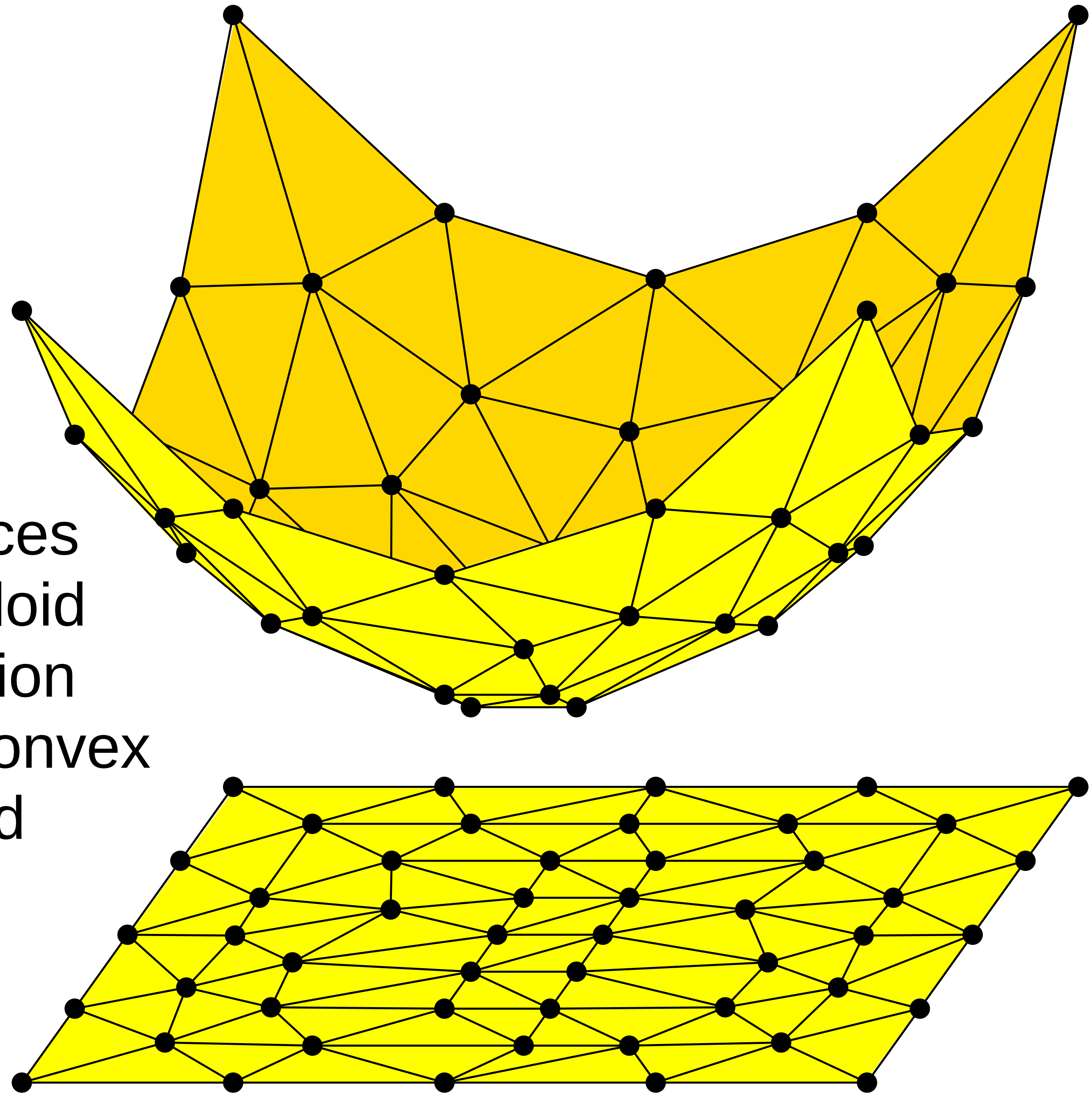




Sometimes we can.

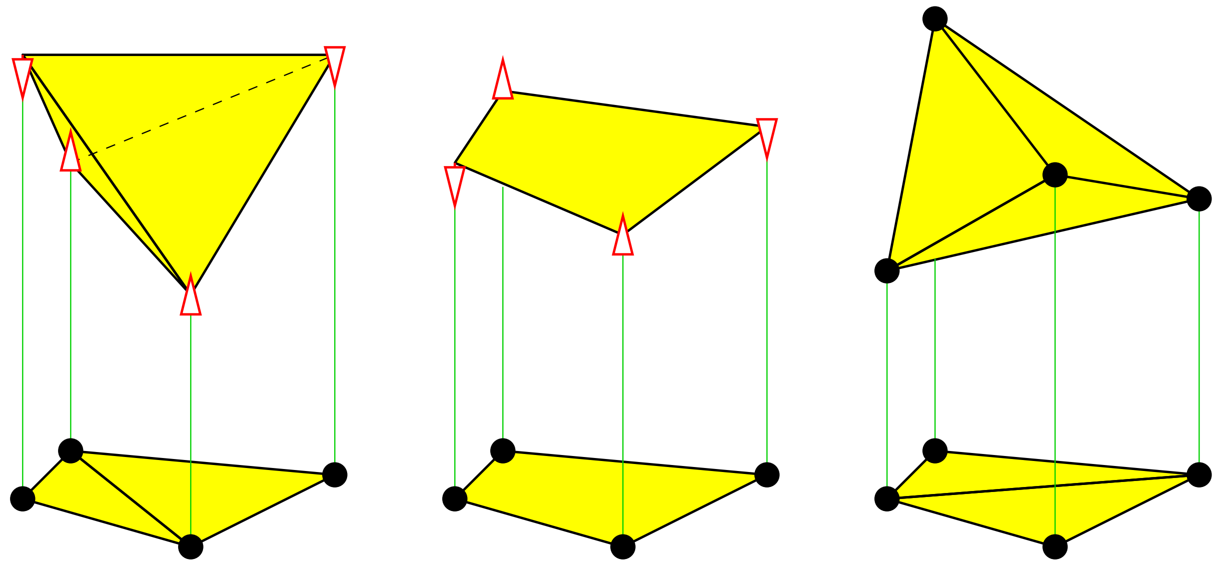
Sometimes we can't.

Parabolic Lifting Map



"Lift" the vertices onto a paraboloid in one dimension higher. The convex hull of the lifted vertices gives the Delaunay triangulation.

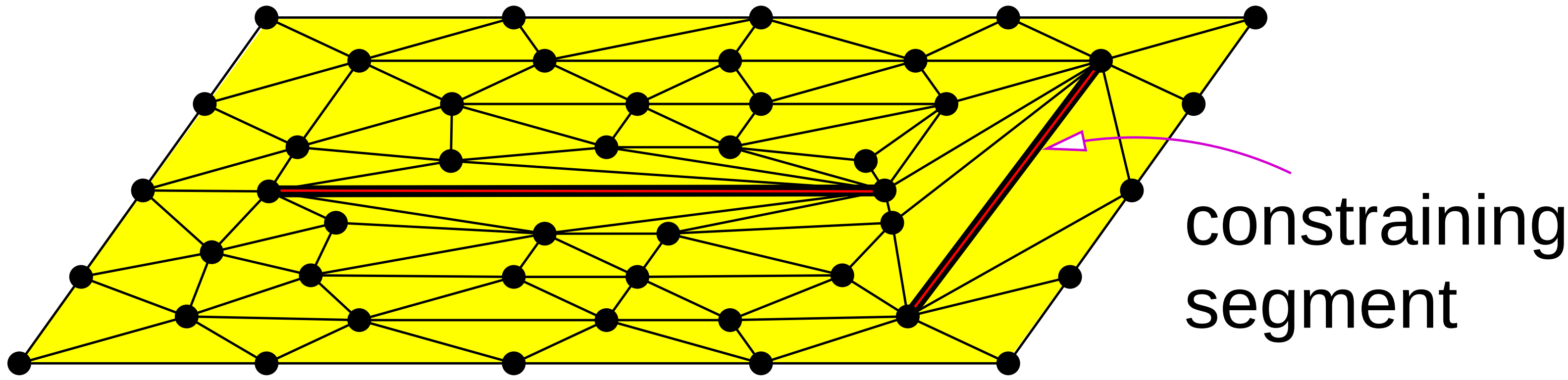
Kinetic Convex Hull



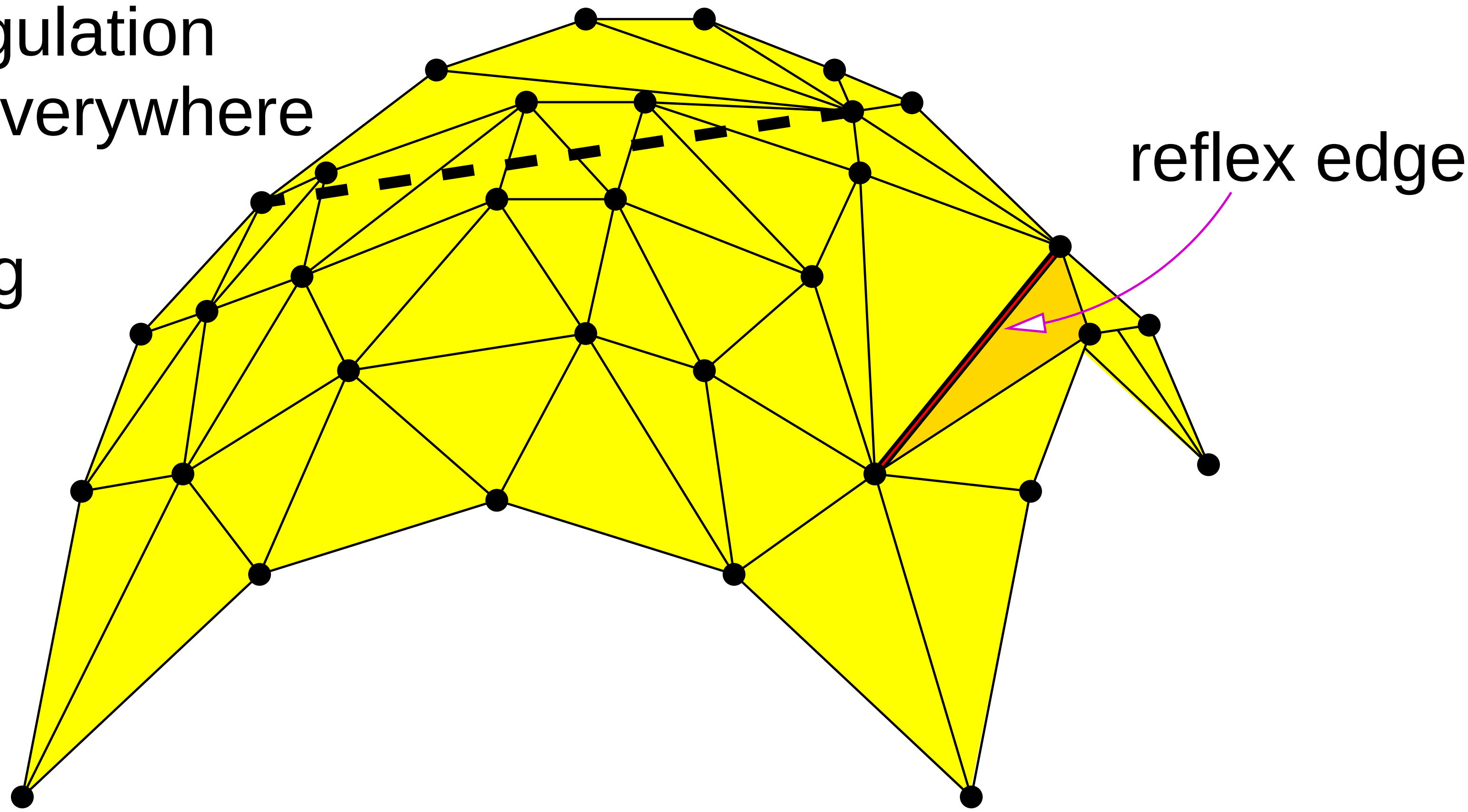
As the lifted vertices move vertically, use flips to maintain the lower convex hull.

Insight: Because a convex hull always exists, the flips cannot get stuck.

A Lifted CDT

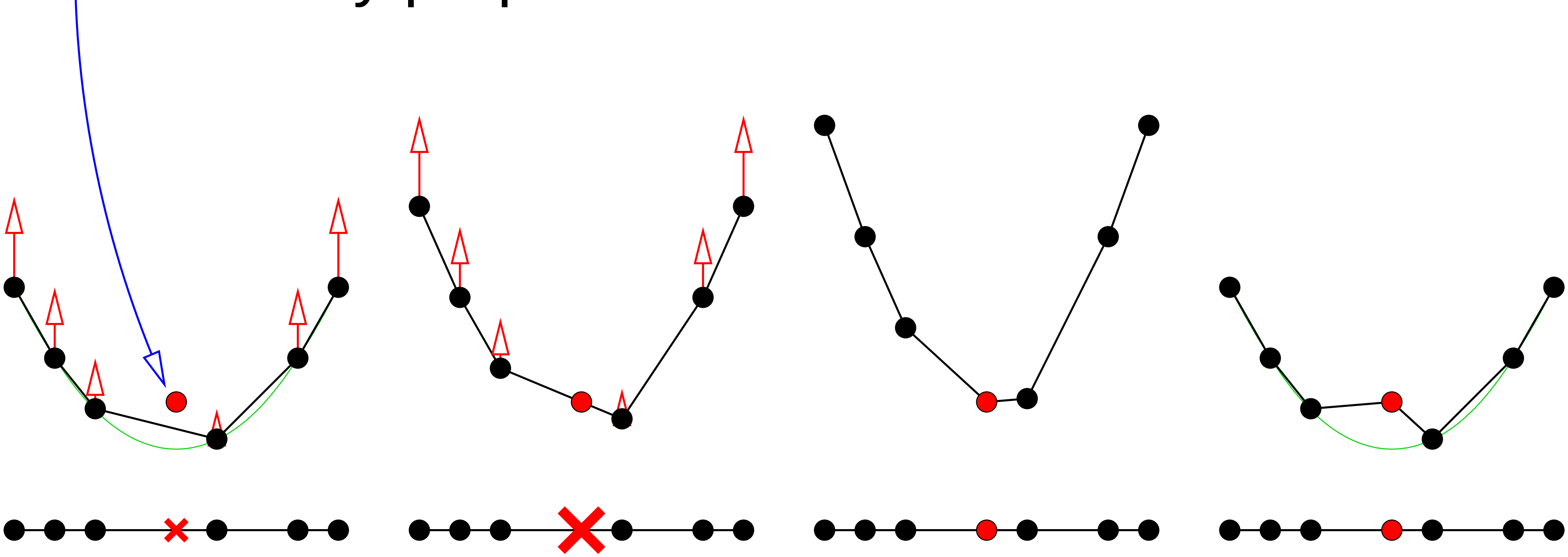


Lifted triangulation is convex everywhere except at constraining segments.



Flip Algorithm for Facet Recovery (1D)

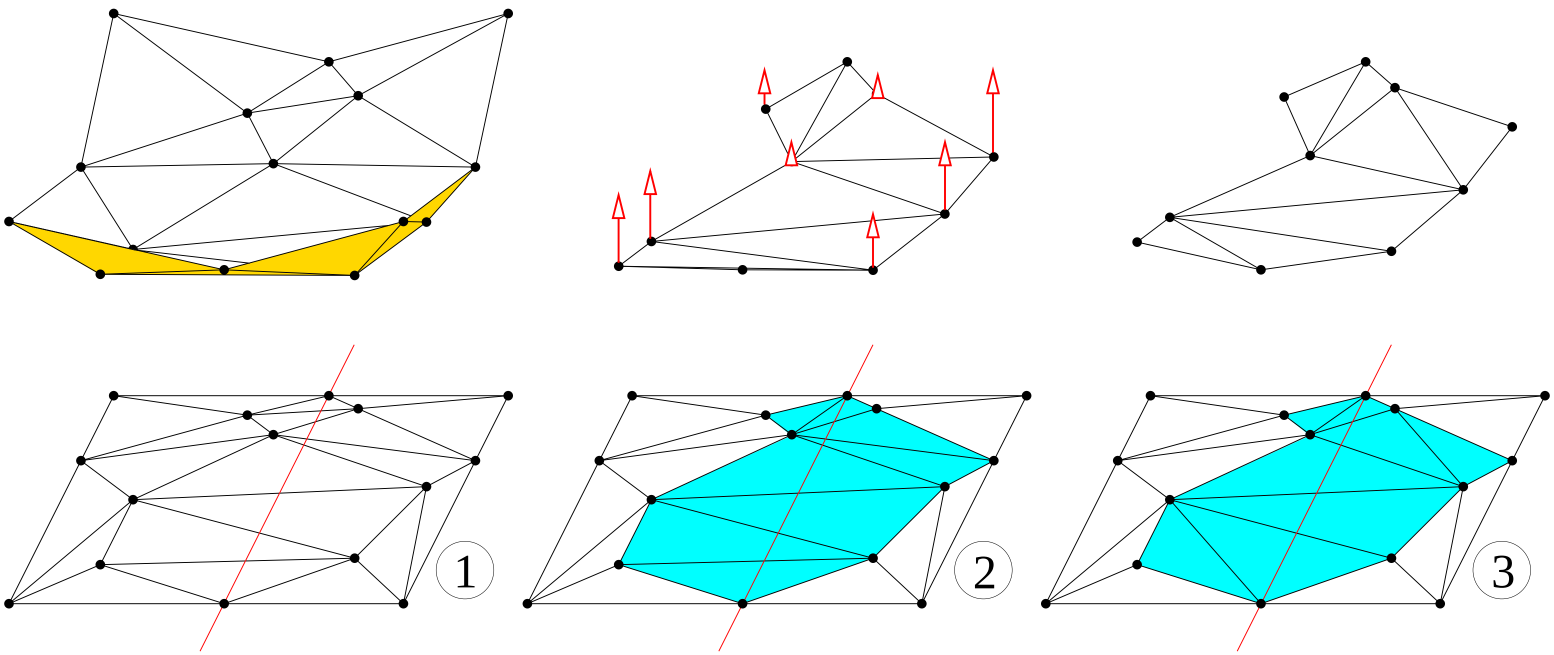
Let’s insert this “facet”! Lifted vertices rise at
a velocity proportional to distance from “facet.”

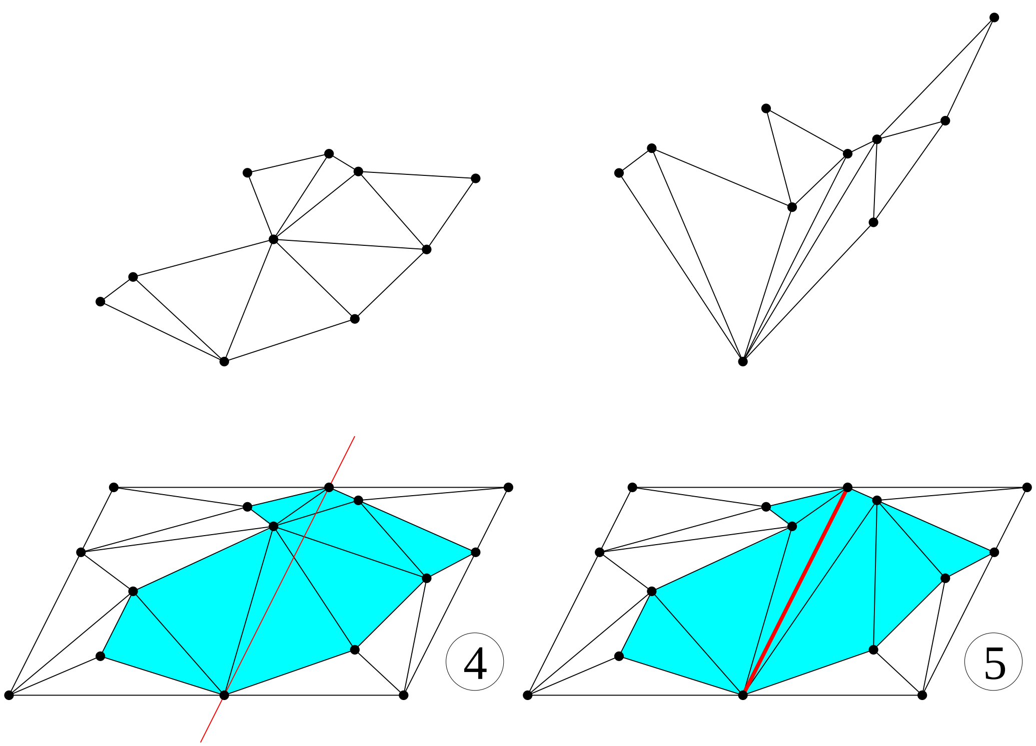


1D triangulation

CDT!

Flip Algorithm for Facet Recovery



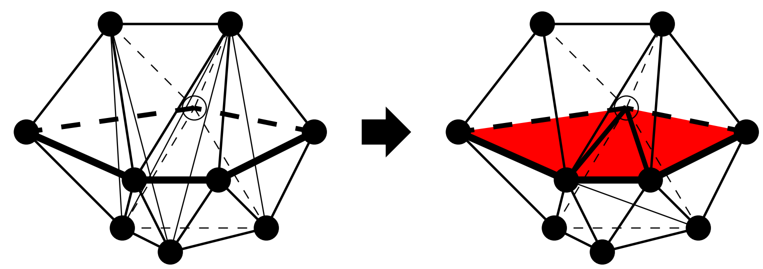


Slowly lift vertices according to distance from new segment. Maintain lower convex hull as we go, using bistellar flips.

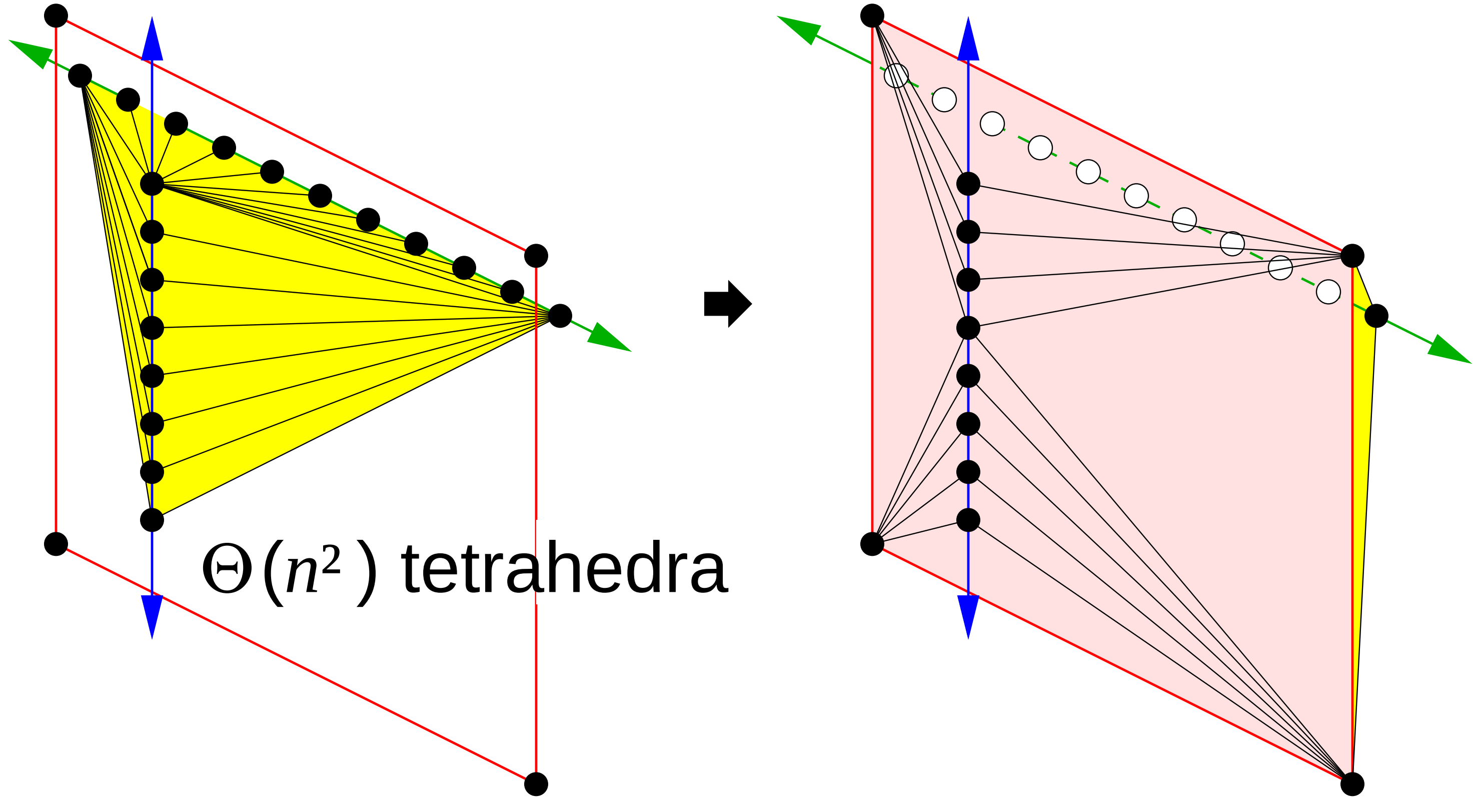
Flip−Based CDT Construction Run Time

Worst−case time to recover one facet:

$O(n^2 \log n)$



Running Time: Lower Bound



Any algorithm might have to do $\Omega(n^2)$ work.

Flip−Based CDT Construction Run Time

Worst−case time to recover one facet:

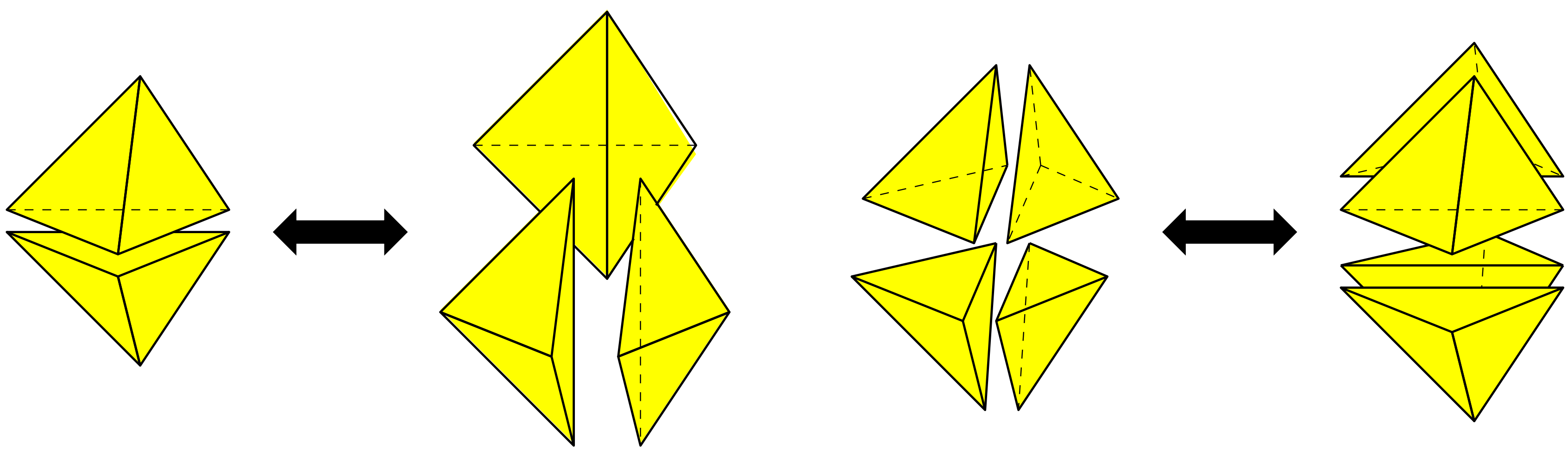
$O(n^2 \log n)$

Worst−case time to recover *any number* of facets and construct a CDT:

$O(n^2 \log n)$

Running Time: Upper Bound

Every bistellar flip either deletes or creates an edge.



Once deleted, an edge can never reappear.
At most n^2 edges can ever appear or disappear.
Therefore, at most $\Theta(n^2)$ flips occur.
Each flip costs $O(\log n)$ time for priority queue handling.

Flip−Based CDT Construction Run Time

Worst−case time to recover one facet:

$$O(n^{\lfloor d/2 \rfloor + 1} \log n)$$

Worst−case time to recover *any number* of facets and construct a CDT:

$$O(n^{\lfloor d/2 \rfloor + 1} \log n)$$

More typical time for many domains:

$$O(n \log n)$$

Conclusions

- Requires fewer new vertices than the other options.
- Optimal for minimizing the worst−case bound on the interpolation error $\| f - g \|_\infty$.
- Works in concert with Delaunay refinement algorithms to offer provably good mesh generation.
- Offers guaranteed lower bounds on edge lengths.